VŠB - Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Monitorování a řízení procesů s využitím technologie .NET

Using .NET Technology for Process Control and Monitoring

Student: Bc. David Plandor

Vedoucí diplomové práce: Ing. Marek Babiuch Ph.D.

Ostrava 2010

Místopřísežné prohlášení studenta

Prohlašuji, že jsem celou diplomovou práci včetně příloh vypracoval samostatně pod vedením vedoucího diplomové práce a uvedl jsem všechny použité podklady a literaturu.

V Ostravě ....

podpis studenta

Prohlášení o použití výsledků práce

Prohlašuji, že

- jsem byl seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo.
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen „VŠB-TUO“) má právo nevýdělečně ke své vnitřní potřebě diplomovou práci užít (§ 35 odst. 3).
- souhlasím s tím, že diplomová práce bude v elektronické podobě uložena v Ústřední knihovně VŠB-TUO k nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že údaje o kvalifikační práci budou zveřejněny v informačním systému VŠB-TUO.
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.
- bylo sjednáno, že užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).
- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě :..... .....
podpis

Jméno a příjmení autora práce: Bc. David Plandor

Adresa trvalého pobytu autora práce: Dolní 397, 742 66 Štramberk

ANOTACE DIPLOMOVÉ PRÁCE

PLANDOR, D. *Monitorování a řízení procesů s využitím technologie .NET : diplomová práce.* Ostrava : VŠB – Technická univerzita Ostrava, Fakulta strojní, Katedra automatizační techniky a řízení, 2010, 61 s. Vedoucí práce: Babiuch, M.

Diplomová práce se zabývá návrhem a realizací softwarové aplikace pro měření a regulaci. Aplikace je určena pro náhradu současného systému pro monitorování a regulaci procesů v teplovzdušném modelu. V úvodu práce byla detailně analyzována aktuálně používaná aplikace, byly stanoveny její přednosti a nedostatky. Na tomto základě byl proveden návrh nové aplikace, byl vypracován její funkční a databázový model s ohledem na možnost obecného využití aplikace s různými hardwarovými zařízeními. Výsledná aplikace WinCoMeT je vytvořena ve vývojovém prostředí Visual Studio 2008 s podporou .NET Framework, pro komunikaci s hardwarovým zařízením využívá komponenty, které lze k aplikaci přidávat využitím dynamické kompilace. Program WinCoMeT byl nasazen do provozu, byly provedeny drobné úpravy a upřesnění algoritmu. Bylo provedeno srovnání s předchozí aplikací. Ovládání a programový kód je dokumentován v textu diplomové práce.

ANNOTATION OF MASTER THESIS

PLANDOR, D. *Using .NET Technology for Process Control and Monitoring : Master Thesis.* Ostrava : VŠB –Technical University of Ostrava, Faculty of Mechanical Engineering, Department of Control Systems and Instrumentation, 2009, 61 p. Thesis head: Babiuch, M.

The master thesis is dealing with control and measuring software application design and development. The application is intended for substitution of currently used software for controlling and measuring data from a hot air model. At the beginning the current application is analyzed at full detail and its pros and cons are determined. On the basis of analysis functional and database models are designed with focus on general use with various hardware devices. The final WinCoMeT application is developed in the Visual Studio 2008 with .NET framework support. For communication purposes WinCoMeT uses components which can be assigned during the runtime due to dynamic compilation. WinCoMeT is implemented, detailed adjustments and algorithm specifications are further done. The newly created application is documented in the master thesis text.

Obsah

1 Úvod 10

2 Hardware – Jednotka CTRL a teplovzdušný model 12

2.1 Komunikační jednotka CTRL V3 12

2.1.1 Základní charakteristika jednotky CTRL a připojení k počítači 12

2.1.2 Komunikace jednotky a počítačem 13

2.1.3 Funkce jednotky CTRL V3 14

2.2 Teplovzdušný model 15

2.2.1 Zapojení modelu 15

2.2.2 Připojení modelu k jednotce CTRL 17

3 Současné softwarové řešení – WinCTRL v2 18

3.1 Základní charakteristika programu WinCTRL v2 18

3.2 Základní nedostatky programu 19

4 Analýza nové softwarové aplikace 21

4.1 Funkční model aplikace 21

4.2 Databázový model aplikace 23

4.2.1 Databáze Microsoft SQL Compact 23

4.2.2 Návrh databáze 24

5 Nástroje pro realizaci programové aplikace 26

5.1 Visual Studio 2008 a .NET Framework 3.5 26

5.1.1 Princip běhového prostředí 26

5.1.2 Klíčové vlastnosti 27

5.1.3 Typy aplikací 28

5.2 Programovací jazyk C# 28

5.2.1 Vlastnosti jazyka 29

5.2.2 Vývojová prostředí 29

5.3 Tvorba komponent ve Visual Studio 2008 a C# 30

5.3.1 Komponenty v paměti počítače 31

5.3.2 Výhody .NET komponent 32

5.3.3 Postup při tvorbě komponent 33

5.3.4 Použití komponent v klientských aplikacích 34

5.4 Dynamická kompilace ve Visual Studiu 2008 36

5.5 Prezentace měřených dat *37*
5.5.1 Microsoft Chart Control 37
5.5.2 Implementace pro použití v prostředí Visual Studio 39
6 Softwarová aplikace WinCoMeT **40**
6.1 Základní informace *40*
6.2 Ovládání programu *41*
6.3 Obecná nastavení programu *42*
6.4 Nastavení vstupů a výstupů *43*
6.5 Měření a regulace *44*
6.6 Export – import měřených dat a parametrů *46*
6.7 Nová zařízení a regulátory *48*
7 Komponenty pro zařízení **51**
8 Srovnání aplikací **53**
9 Závěr **54**
10 Použitá literatura **56**
11 Přílohy **59**
Příloha A – Zapojení pinů konektoru jednotky CTRL *59*
Příloha B – Srovnání obdobných měření ve WinCTRL a WinCoMeT *60*

Seznam zkratek

.NET – (čti *dotnet* podle anglického *dot NET* = tečka NET) – NET pochází ze slova network = síť – zastřešující název pro soubor technologií v softwarových produktech, které tvoří celou platformu.

ASCII – anglická zkratka pro *American Standard Code for Information Interchange*, tedy americký standardní kód pro výměnu informací.

CAD – *Computer* **–** *Aided Design*, počítačem podporované projektování, nebo míněno na obecný CAD systém jako *Computer-Aided Drafting* **–** počítačem podporované kreslení.

CNC – *Computer Numerical Controlled* – číslicové ovládání strojů řízené počítačem.

COM – původní, stále běžně používaný název rozhraní sériového portu na IBM PC kompatibilních počítačích.

CR – *Carriage Return* – ukončovací znak řádku a ASCII kódem 13.

CLR – *Common Language Runtime* **–** společné běhové prostředí v platformě .NET.

CSV – *Comma Separated Values* **–** hodnoty oddělené čárkami – jednoduchý souborový formát určený pro výměnu tabulkových dat.

CTS – *Common Type System* – společný typový systém. Zajišťuje typovou bezpečnost. Definuje základní typy, které musí každý jazyk platformy .NET podporovat.

DLL – *Dynamic* **–** *Link Library* **–** dynamická knihovna **–** programový modul, který může být sdílen více programy. Operační systém načítá do operační paměti kód vlastního programu a také kód dynamické knihovny, kterou program vyžaduje ke své činnosti.

LF – *Line Feed* – ukončovací znak řádku a ASCII kódem 10.

MDI – *Multiple Document Interface* – aplikace s jedním hlavním (*parent*) formulářem a dalšími podřízenými formuláři s možností vzájemné komunikace.

MSIL – *Microsoft Intermediate Language* – jazyk relativních adres v platformě .NET.

NC – *Numerical Control* – číslicové ovládání strojů.

OLE – *Object Linking and Embedding* – soubor protokolů, rozhraní a funkcí. Umožňuje aplikaci poskytovat svou funkcionalitu nebo řídit funkcionalitu jiných aplikací na jednom počítači i mezi počítači v síti.

OOP – *Object Oriented Programming* – objektově orientované programování.

RS-232 – **s**tandard, resp. jeho poslední varianta RS-232C z roku 1969, také sériový port nebo sériová linka, se používá jako komunikační rozhraní osobních počítačů a další elektroniky.

SQL (čti *es-kjů-el* nebo také *síkvl*) – *Structured Query Language* – strukturovaný dotazovací jazyk - standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích.

USB – *Universal Serial Bus* – je univerzální sériová sběrnice. Moderní způsob připojení periférií k počítači.

W3C – *World Wide Web Consortium* – je mezinárodní konsorcium, jehož členové společně s veřejností vyvíjejí webové standardy.

WinCoMeT – *Controlling and Measuring Tool for Windows* – je název nově vytvořené aplikace pro měření a regulaci pro operační systémy Windows.

XML – *Extensible Markup Language* – rozšiřitelný značkovací jazyk **–** obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C.

1 Úvod

Monitorování a řízení procesů je obor, který se již řadu let velmi rozvíjí. V počátcích automatizace bylo zapojení počítačů prioritní především ve výrobním sektoru průmyslu. První číslicově řízený stroj byl navržen již v 50. letech 20. století. Realizace těchto návrhů a větší rozvoj počítačově řízených systémů však přišel až v 70. letech. V následujícím desetiletí již bylo široce využíváno NC a CNC strojů, s nimiž byl úzce spjat návrh pomocí systémů CAD. Dnes již počítače ovládají všechny organizační úrovně podniků. Výroba je řízena počítačovými automatizovanými operátorsko-vizualizační systémy, data jsou ukládána do databází, manažerské aplikace umožňují plánovat, vytvářet statistiky, designérské programy usnadňují vývoj nových produktů.

Velmi významnou oblastí vývoje nových technologii je modelování systémů. Lze tak s podstatně nižšími náklady v laboratorních podmínkách ověřit fyzikální principy, které jsou pak využity při návrhu reálných zařízení. Takovýmto modelem je např. model teplovzdušné soustavy využívaný v praktických cvičeních na katedrách automatizace v České republice. Tento model komunikuje s počítačem PC prostřednictvím komunikační jednotky CTRL. Vlastní monitorování a řízení soustavy pak zajišťuje softwarová aplikace WinCTRL.

Cílem této práce je vytvořit zcela nový softwarový systém, který nahradí současnou aplikaci WinCTRL. Pro nové řešení bude opět využita jednotka CTRL, bude pro ni vytvořena komponenta na platformě .NET, která bude řešit veškeré komunikace mezi jednotkou a klientskými aplikacemi. Prezentační část systému bude oddělena a bude využívat prostředků poskytovaných komponentou. Takto bude možno do budoucna upravovat pouze požadovanou část (jen komunikační nebo jen prezentační) nebo také propojovat jednotlivé části s jinými aplikacemi. Např. komponenta může poskytovat data pro webové rozhraní nebo prezentační aplikace může využívat komponentu podporující komunikaci s jiným hardwarovým zařízením.

V první fázi práce bude potřeba navrhnout komplexně celý systém, vytvořit funkční a datový model, specifikovat požadované funkce systému. Pak bude možno přejít k volbě vhodného vývojového nástroje a ověření jeho možností. Ověření bude spočívat v otestování potřebných elementárních funkcí jako sériová komunikace s jednotkou CTRL u komponenty nebo nalezení a otestování vhodných komponent pro vykreslování grafů.

Bude-li pravděpodobné, že zvolené prostředí a přidružené komponenty budou splňovat vytyčené cíle, lze přejít k vlastní tvorbě programového kódů aplikace. Realizovanou aplikaci bude nutno implementovat a důkladně otestovat všechny její funkce. V závěru práce bude vhodné obě aplikace přehledně srovnat a navrhnout další vývoj nové aplikace.

2 Hardware – Jednotka CTRL a teplovzdušný model

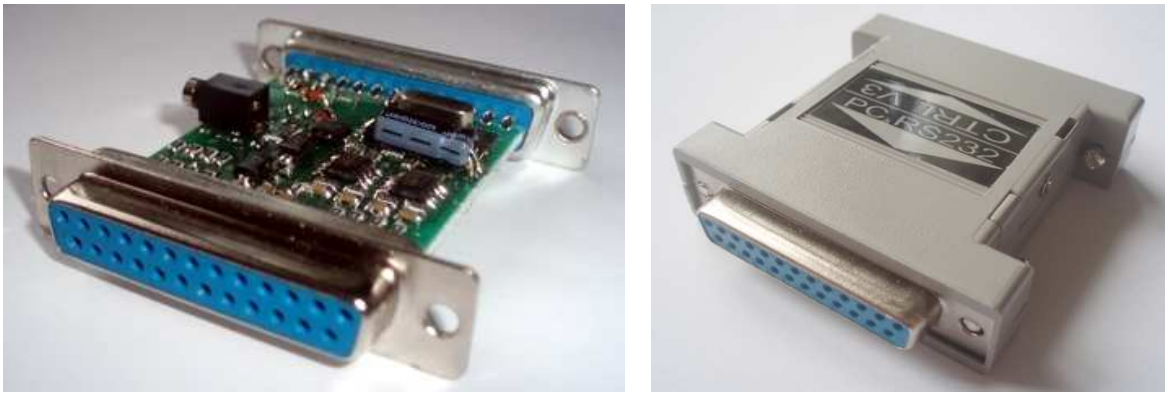
Nejprve popíšeme hardwarová zařízení, která jsou použita u současného řešení a budou využita také pro nové softwarové řešení. Jedná se o komunikační jednotku CTRL a teplovzdušný model.

2.1 Komunikační jednotka CTRL V3

Hardwarové řešení zůstane zachováno, bude tedy použita opět jednotka CTRL V3, avšak softwarově se k ní bude přistupovat odlišným způsobem. Není tedy vyloučeno do budoucna vytvořit komponentu pro jinou komunikační jednotku a k ní použít již hotovou prezentační část.

2.1.1 Základní charakteristika jednotky CTRL a připojení k počítači

Jednotka CTRL V3 je již třetí generací tohoto komunikačního zařízení.



Obrázek 2-1 - Komunikační jednotka CTRL V3

Jednotka obsahuje

- 4 analogové vstupy (0-10V)
- 2 analogové výstupy (0-10V, 50mA)
- 4 digitální vstupy
- 4 digitální výstupy

Digitální výstupy jsou přizpůsobeny k přímému ovládání 12V relé. Jednotka je oproti předchozím generacím výrazné miniaturizovaná, je umístěna do standardního sériového krytu mezi dvěma konektory CANON 25. Jeden z těchto konektorů slouží pro připojení

vstupů a výstupů a druhý pro sériovou linku RS232, která má opto-elektrické oddělení. Pro komunikaci s počítačem PC postačí pouze připojit k standardnímu sériovému portu COM a není potřeba instalovat jakékoli ovladače. V dnešních počítačích již většinou nenajdeme COM porty a je tedy potřeba využít převodník. Nejčastěji se používá převodník USB-RS232, pak ovšem je nutno nainstalovat ovladače určené pro tento převodník. Jednotku je nutno napájet z univerzálního nestabilizovaného zdroje 12V s minimálně 300mA. [Klán, 2003]

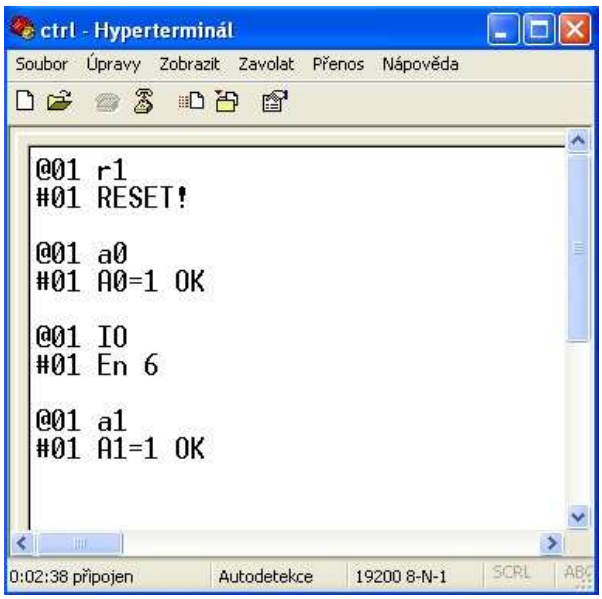
2.1.2 Komunikace jednotky a počítačem

Používáme-li standardní COM port počítače, můžeme ihned ověřit komunikaci např. pomocí programu *Hyperterminál*, který je součástí příslušenství operačních systémů Windows. Při využití převodníku USB-RS232 je potřeba u ovladače nastavit číslo COM portu, který bude pro komunikaci použit. Pak již je ve správci zařízení systému Windows COM port přítomen a je možno jej používat úplně stejně jako v případě hardwarového COM portu.

Sériová komunikace musí být nastavena následujícím způsobem:

- 19200 Baud
- 8 datových bitů
- Bez parity
- 1 stop bit
- DTR (*Data Read*) ve stavu ON (chceme-li data také přijímat)

Komunikační protokol jednotky je znakový (7 bit US-ASCII) a využívá ukončovací znak (terminátor) CRLF, což je kombinace znaků CR (*Carriage Return*) a LF (*Line Feed*). K vysílaným sekvencím znaků se terminátor přidává automaticky. Jednotka CTRL V3 odpovídá na každý příkaz potvrzovací zprávou nebo chybovým hlášením. Každý příkaz pro jednotku musí začínat znaky „@01“, za nimi následuje vlastní příkaz a ten je zakončen terminátorem. Odpověď jednotky začíná znaky #01, pak následují výstupní data nebo chybové hlášení a celý kód ukončuje opět terminátor. Komunikaci jsme ověřili v programu *Hyperterminál*. Na obrázku 2-2 je zobrazeno okno terminálu s několika zadanými příkazy, za kterými následují odpovědi jednotky.



Obrázek 2-2 - Ověření komunikace v programu Hyperterminál

2.1.3 Funkce jednotky CTRL V3

Ovládání výstupů

- Nastavení a zapnutí či vypnutí digitálních výstupů.
- Zjištění stavu digitálních výstupů.
- Podmíněný časový bitový výstup – při sepnutém stavu digitálního vstupu nebo po zadání příkazu je možno na určitou dobu sepnout digitální výstup. Při modifikaci příkazu lze vypnout čítač nebo zajistit udržení aktuální hodnoty digitálního výstupu. Také lze zjistit stav tohoto sestupného čítače.
- Generování impulsů na digitálním vstupu. Pomocí parametrů lze zadat počet impulsů, jejich šířku, šířku mezery a koncový stav.
- Nastavení analogových vstupů.

Načítání vstupů

- Zjištění stavu digitálních vstupů. Lze také zjistit existenci impulsů 1-0-1 nebo 0-1-0. Vždy se zjišťují pouze změny od posledního čtení.
- Vstupy čítačů lze volit programově z digitálních vstupů. Lze zjistit stav čítače popř. nastavit jeho hodnotu.
- Čtení analogových vstupů.

Operace s pamětí

Čtení a zápis do paměti EEPROM. Nastavují se různé parametry, které mají vliv na chování jednotky CTRL V3.

- Povolení či zakázání automatického vysílání změn digitálních vstupů.
- Parametry kalibrace analogových vstupů.
- Přednastavení stavu digitálních vstupů po resetu.
- Hodnoty čítačů.
- Zdrojové vstupy pro čítače.
- Nastavení periody automatického vysílání stavu vstupů a výstupů a další.

2.2 Teplovzdušný model

Jde o laboratorní model určený pro výuku regulace. Lze v něm nasimulovat a demonstrovat základní regulační pochody. Obsahuje softwarové řešení regulátorů (dvoupolohové regulátory, PI, PID, fuzzy řízení a další). Tento model lze připojit k počítači pomocí multifunkční karty, která disponuje analogovými vstupy a výstupy. Programová aplikace WinCTRL pak zajišťuje zobrazení měřených dat, regulaci a generování signálů. Kromě programu WinCTRL lze použít také program v prostředí Matlab.

2.2.1 Zapojení modelu

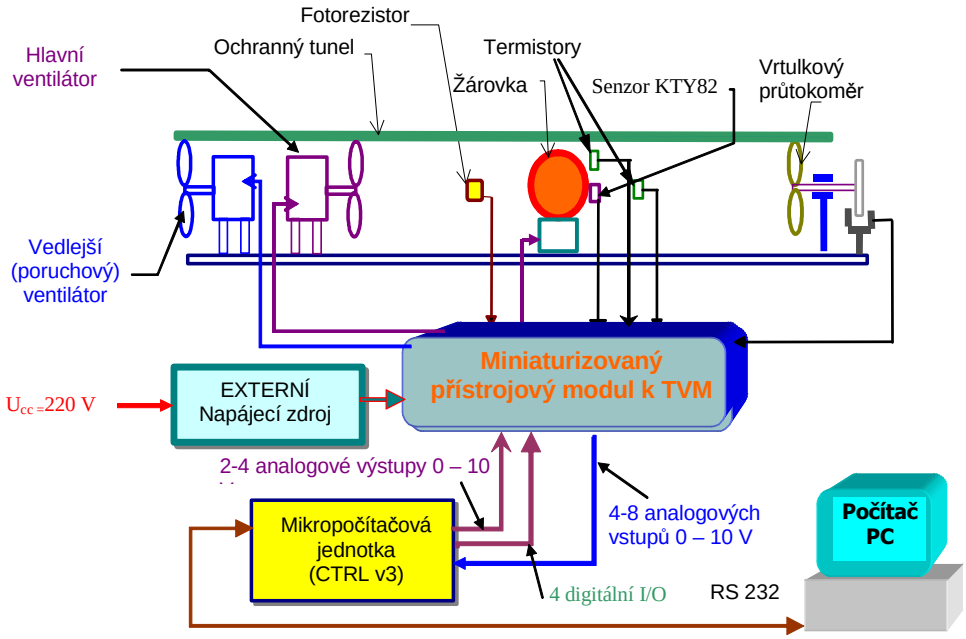
Model teplovzdušného obvodu je tvořen žárovkou napájenou z řiditelného zdroje napětí (vytváří tepelný a světelný zdroj), jež je umístěná v krytém tunelu, kterým proudí vzduch vznikající činností ventilátoru (ten je rovněž napájen pomocí řiditelného zdroje napětí). [Smutný, 2007]

V tunelu je umístěno několik snímačů:

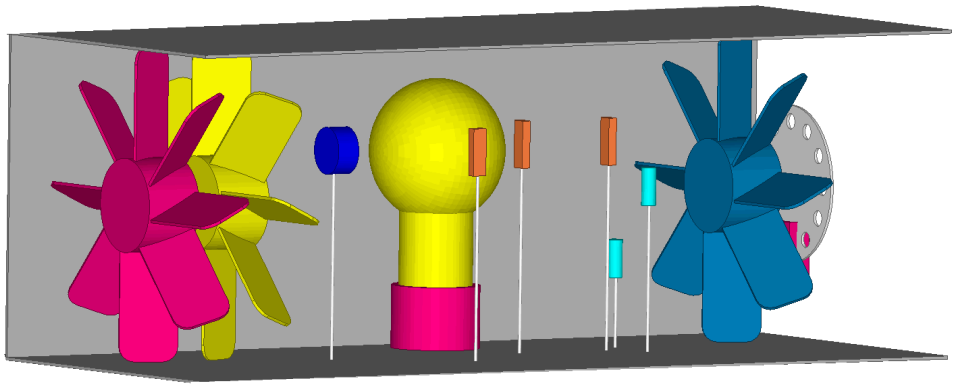
- **Tři snímače teploty** - termistor T_3 měřící teplotu baňky žárovky, T_2 - teplota vzduchu v bezprostřední blízkosti baňky žárovky, T_1 - teplota vzduchu v zadní části tunelu.
- **Fotodetektor** - fotorezistor FR_1 měřící jas žárovky.

- **Termoanemometr** - je tvořen dvěma termistory, první je umístěn v tunelu a měří rychlost proudění vzduchu - TA, druhý referenční termistor RT není proudem vzduchu ovlivňován.
- **Objemový vrtulkový průtokoměr** - VP - měřicí vrtulka s připojeným snímačem otáček.

Na obrázku 2-3 je zobrazeno schéma a zapojení teplovzdušného modelu, na obrázku 2-4 potom jeho trojrozměrná vizualizace.



Obrázek 2-3 - Zapojení modelu teplovzdušného obvodu s počítačem PC a jednotkou CTRL [Smutný 2007]



Obrázek 2-4 - Teplovzdušný model (TVM) - 3 D vizualizace [Smutný 2007]

2.2.2 Připojení modelu k jednotce CTRL

Mikropočítačová jednotka CTRL slouží pro připojení vnějších signálů ze zkoumaných fyzických objektů k PC (např. teplovzdušného modelu TVM). Pomocí konektoru je možno připojit až 16 analogových vstupů a 4 analogové výstupy.

Tabulka 2-1 – Používané analogové vstupy jednotky CTRL - měřené veličiny z teplovzdušného modelu

Analogové vstupy	
Vstup 1	snímač jasu žárovky (fotorezistor) F
Vstup 2	snímač teploty v blízkosti baňky žárovky T_1
Vstup 3	snímač teploty baňky žárovky T_2
Vstup 4	snímač teploty na výstupu z tunelu T_3
Vstup 6	termoanemometr TA
Vstup 7	vrtulkový průtokoměr VP

Tabulka 2-2 – Používané analogové výstupy jednotky CTRL - řídicí veličiny teplovzdušného modelu

Analogové výstupy	
Výstup 1	ovládací napětí na žárovce
Výstup 2	ovládací napětí na hlavním ventilátoru (řízení otáček) V_1
Výstup 3	ovládací napětí na vedlejším ventilátoru (řízení otáček) V_2

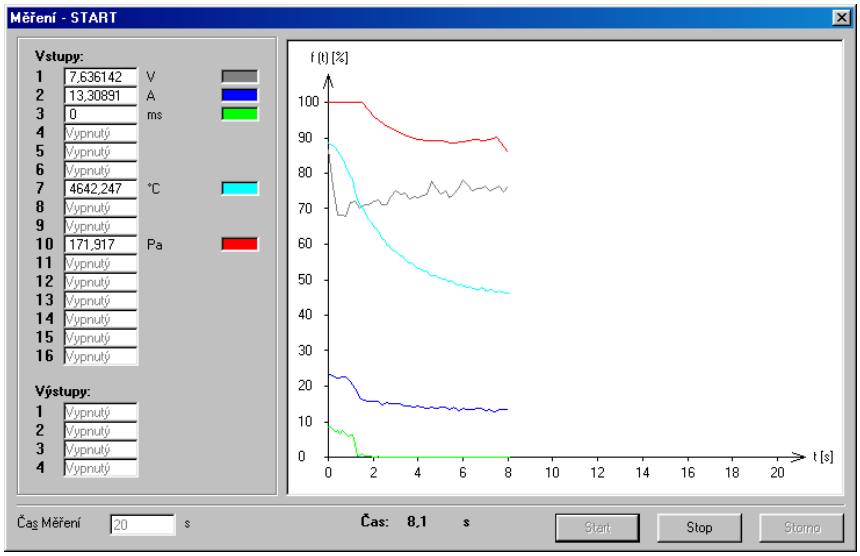
3 Současné softwarové řešení – WinCTRL v2

Nejprve budeme charakterizovat současnou programovou aplikaci WinCTRL verze 2, která se používá pro zobrazení měřených dat z teplovzdušného modelu a také pro regulaci. Tuto aplikaci máme nahradit novým řešením, bude tedy nutné detailně prověřit všechny jeho vlastnosti, analyzovat případné nedostatky, kterých je nutno se vyvarovat při návrhu a realizaci nové podoby programu.

3.1 Základní charakteristika programu WinCTRL v2

Program WinCTRL byl vytvořen v prostředí Visual Studio 6 v programovacím jazyce Visual Basic. S vývojem nových operačních systémů bylo nutno často zasahovat do kódu programu a upravovat jej dle změn. Ukončovací znaky (CR, LF) v sériové komunikaci byly řešeny jinak ve Windows 98 a NT než např. ve Windows XP, proto muselo existovat několik verzí programu pro různé operační systémy. Do programového kódu bylo proto nutno často zasahovat. Na úpravách aplikace se podílelo více autorů, což je zřejmé z programového kódu, který je v některých místech nepřehledný, některé funkční celky se vyskytují duplicitně. Velkou výhodou při analýze programového kódu byly detailní komentáře. Grafický vzhled aplikace odpovídá standardu prostředí Visual Studio 6.

Program obsahuje všechny funkčnosti – řeší komunikaci s jednotkou CTRL, grafické zobrazení výstupních dat, slouží k zadávání vstupních údajů uživatelů atd. Pokud se tedy provede změna v komunikaci, je třeba nahradit celý program, což považuji za jeho největší nevýhodu.



Obrázek 3-1 – Měření v aplikaci WinCTRL v2 [Smutný, 2007]

Program pracuje pouze s jedním zařízením a tím je jednotka CTRL, programový kód obsahuje také komunikaci s převodníkem AD512, tato komunikace však byla po úpravách z důvodu přechodu na systém Windows XP v aplikaci zakázána.

Nyní shrneme základní nedostatky programu, které by měla vyřešit nově navržená verze programu.

3.2 Základní nedostatky programu

- **Nastavení sériové komunikace**

  V programu lze nastavit pouze port COM1 a COM2. Moderní počítače již neobsahují COM porty a je nutno pro sériovou komunikaci použít převodník USB-COM. V počítači se vytvoří nový virtuální COM port. Číslování těchto portů je variabilní, proto nelze spoléhat na pevně zadané porty.

- **Rozdíly v sériové komunikaci dle operačního systému**

  Program byl dříve provozován na operačním systému Windows 2000, kde pracoval správně. Po přechodu na systém Windows XP nebyla sériová komunikace funkční a bylo nutno upravit programový kód. Existovaly pak dvě verze programu pro každý operační systém a obě verze bylo nutno paralelně udržovat (každá změna musela být provedena dvakrát).

- **Každá změna v kódu vyžaduje kompilaci celého programu**

  Jakákoli změna v programu vedla k jeho kompletní nové kompilaci a vzniku nové verze programu. Nebylo možné např. upravit jen komunikační část a tu nahradit zatímco by prezentační část programu zůstala původní.

- **Prezentace měřených dat**

  Po skončení měření zmizí graf a zobrazí se tabulka změřených dat. Není možné prohlížet zároveň změřená data a graf.

- **Připojení jiných zařízení**

  Nelze připojit jiné zařízení bez modifikace programu. Nyní lze použít pouze jednotku CTRL přes sériové rozhraní. Pokud by bylo potřeba připojit jiné zařízení, musel by se upravit programový kód, a následně vytvořit nová verze programu.

- **Nelze měnit parametry během měření**

  Během měření nelze měnit parametry vstupů a výstupů s výjimkou žádané veličiny regulátorů.

- **Ukládání parametrů uživatele**

  Neukládají se některé uživatelem změněné parametry. Jde např. o barvy pro vykreslování datových řad vstupů a výstupů do grafu.

- **Provoz bez připojeného zařízení**

  Program nelze použít bez připojeného nebo alespoň softwarově simulovaného zařízení. Není možné tedy např. pouze generovat testovací signály.

- **Parametry grafu**

  Nelze měnit parametry grafu. Jde o titulek grafu, popis os, typ grafu apod. Většina těchto údajů je v programu napevno nastavena a není možné je uživatelsky změnit. Nelze měnit velikost grafu při měření resp. velikost formuláře, ve kterém měření probíhá.

- **Export dat**

  Měřená data lze z programu exportovat několika způsoby. Prvním je specifický formát pro program WinCTRL, ze kterého lze opět data načíst. U tohoto formátu existuje omezení na max. 7 vstupů či výstupů se shodnou vzorkovací periodou. Dále je možno provést export v textovém formátu s oddělovači (tabulátory nebo pevnými mezerami).

- **Export grafu**

  Graf lze z aplikace exportovat pouze ve formátu **.bmp*. Jde tedy o bitmapové zobrazení, které není možno dále dynamicky editovat (lze jej upravit pouze jako bitmapový obrázek). Navíc tento obrázek není exportován v podobě, jak je zobrazen v aplikaci. Ačkoli v aplikaci je zobrazen nadpis a popis os, v exportovaném souboru tyto náležitosti chybí.

- **Nefunkční nápověda**

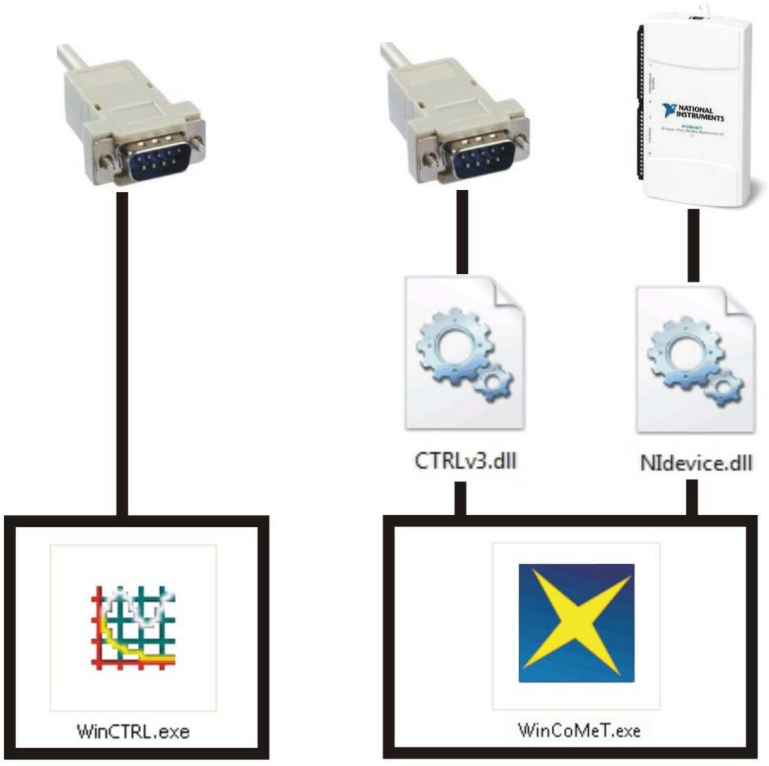
  Není dostupná nápověda k programu WinCTRL. Není mi však jasné, zda je to problém pouze verze aplikace, kterou mám k dispozici.

4 Analýza nové softwarové aplikace

Nejprve rozebereme základní požadavky na novou aplikaci. Sestavíme schéma funkčního modelu. Budeme obecně definovat objekty, jejich vlastnosti a způsob jejich vzájemné komunikace. V druhé části kapitoly se zaměříme na volbu vhodného systému pro uložení dat, zvolený systém popíšeme a navrhneme databázový model – tabulky a vztahy mezi nimi.

4.1 Funkční model aplikace

Základem aplikace bude prezentační program WinCoMeT (zkratka pro *Controlling and Measuring Tool for Windows*). K volbě odlišného názvu pro aplikaci nás nutí odlišné principy obou programů a také hardwarová nezávislost nové aplikace. Program WinCTRL je úzce vázán na jednotku CTRL, WinCoMeT bude využívat prakticky jakoukoli měřicí kartu podporující systém Microsoft Windows.



Obrázek 4-1 - Srovnání modelu aktuálního (vlevo) a nového (vpravo) softwarového řešení

Všechna komunikační zařízení budou připojena prostřednictvím komponent. Bude tedy vytvořen zvláštní programový kód pouze pro komunikaci s jednotkou CTRL a ten bude své metody poskytovat programu WinCoMeT. Pro komunikační komponenty bude stanovena přísná konvence, kterou bude potřeba dodržet při vytváření dalších

komunikačních komponent pro jiná zařízení. Tento systém spolehlivě zajistí možnost dynamického rozvoje aplikace bez nutnosti zasahovat do hlavního programu WinCoMeT. Ze schématu na obrázku 4-1 je zřejmé, že nový model (vpravo) je daleko flexibilnější, jednotlivé části mohou existovat samostatně a mohou být propojeny s jinými na základě předem daných pravidel. V případě změny postačí nahradit pouze upravenou část.

Nejprve bude nutné vytvořit komponentu pro jednotku CTRL, neboť ta je aktuálně používána pro komunikaci s teplovzdušným modelem. V rámci dalšího vývoje aplikace se zaměříme na multifunkční karty od společnosti National Instruments využívané na katedře. Tyto karty umožňují navíc simulaci zařízení bez fyzicky připojené multifunkční karty.

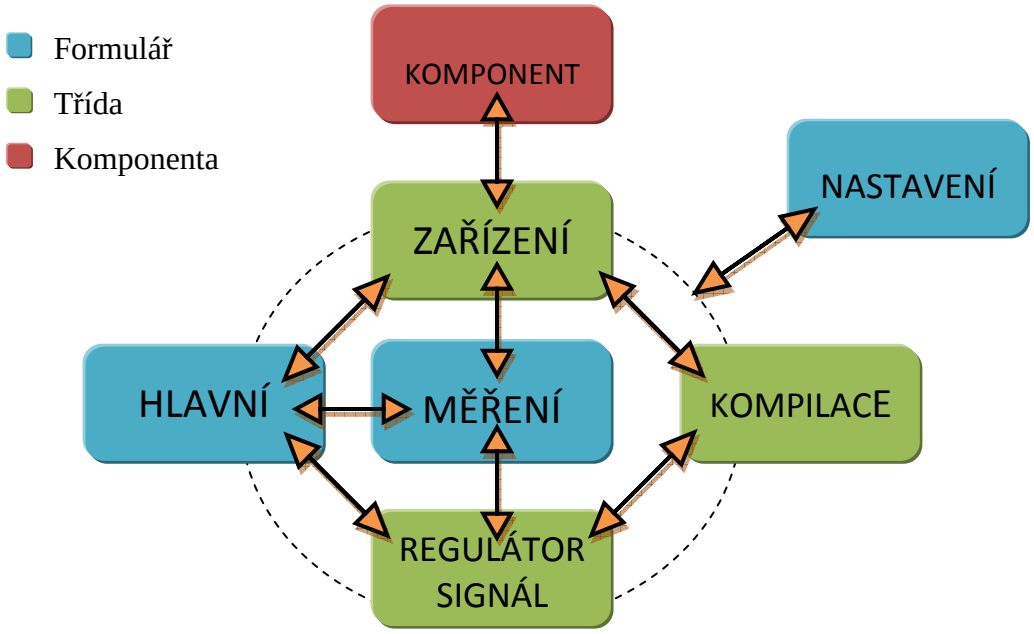
Veškerá programová data budou ukládána do SQL databáze. Prezentace naměřených dat bude realizována pomocí zvolené komponenty. Export dat bude proveden do standardních formátů. Grafy bude možno uložit přímo do souboru jako grafiku, bude možno uložit jej do schránky pro přímé vložení do jiných aplikací a bude také přidána možnost přímého exportu dat do programu Microsoft Excel včetně vykreslení grafu. Provedená měření bude možno uložit do formátu specifického pro program WinCoMeT s možností zpětného otevření, prohlížení a exportu. Veškeré parametry pro generované signály a regulátory bude možno v programu editovat a to včetně vzorců a definic výpočtů. Uživatel se základní znalostí programování tak bude schopen sám aplikaci rozvíjet, přidávat nové regulátory, upravovat nebo vylepšovat již existující. Nově vytvořené skripty a vzorce budou dynamicky kompilovány a budou využívány prostřednictvím standardního volání metod.

Interně bude program navržen pomocí objektů, které spolu budou komunikovat (obrázek 4-2). Bude se jednat o formuláře, třídy a vždy jednu komponentu zařízení. Hlavní formulář bude při spuštění vytvářet instance objektů. Nejprve vytvoří instanci třídy pro zařízení, kde zkompiluje aktuálně zvolenou (resp. poslední v parametrech uloženou) **.dll* komponentu zařízení. Třída pro zařízení pak zprostředkovává komunikaci mezi komponentou a ostatními objekty aplikace, poskytuje metody pro čtení, zápis, inicializaci a nastavení komponenty.

Dále v pořadí je vytvoření instance objektu pro regulátory a signály, zde dochází při inicializaci ke kompilaci metod pro všechny regulátory a signály definované v programu. Regulátory a signály jsou detailně definovány – jsou určeny jejich parametry včetně datových typů, omezení jejich hodnot a je nadefinován skript pro výpočet jejich výstupních hodnot. Právě z důvodu této definice je nutné při spuštění programu z těchto textových

databázových hodnot vytvořit třídu a metody, které budou při měření a regulaci v programu využívány. Pro vlastní kompilaci tříd zařízení, regulátorů a signálů je využívána samostatná třída.

Jsou-li úspěšně vytvořeny instance pro zařízení a regulátory, je možné přistoupit k vytvoření instance formuláře pro měření a regulaci. Pro měření i regulaci existuje pouze jeden formulář, pro měření se pouze skryjí nepotřebné prvky formuláře a jednotlivé módy se náležitě zohlední v programovém kódu. Formulář pro nastavení bude zajišťovat možnost změny parametrů pro dané objekty.



Obrázek 4-2 - Interní model komponent programu WinCoMeT

4.2 Databázový model aplikace

Program WinCoMeT bude využívat SQL databázi pro ukládání parametrů programu, pro veškerá uživatelská nastavení a interní data aplikace. SQL databáze umožňují efektivní rychlou práci s daty, data existují ve standardním formátu SQL tabulek a je možné je zpracovávat i z externích zdrojů (např. pomocí nástrojů prostředí Microsoft SQL Server).

4.2.1 Databáze Microsoft SQL Compact

Základním kritériem pro volbu SQL databáze byla plná podpora ze strany vývojového prostředí Microsoft Visual Studio 2008. Dále byla preferována jednodušší varianta, neboť výsledná aplikace není z pohledu databáze náročná na množství dat a datových struktur.

Při analýze připadaly v úvahu dvě varianty SQL databáze – Microsoft SQL Express a Microsoft SQL Compact. Po porovnání obou databází byla zvolena verze *Compact*, neboť přesně splňuje požadavky pro cílovou aplikaci. Rozhodujícím kritériem byla možnost zapouzdření do cílové aplikace. Po instalaci programu WinCoMeT tak nebude nutno navíc instalovat databázové prostředí. Komponenta databáze pracuje jako in-process, je tedy součástí procesu nadřízené aplikace.

Mezi základní vlastnosti Microsoft SQL Compact patří:

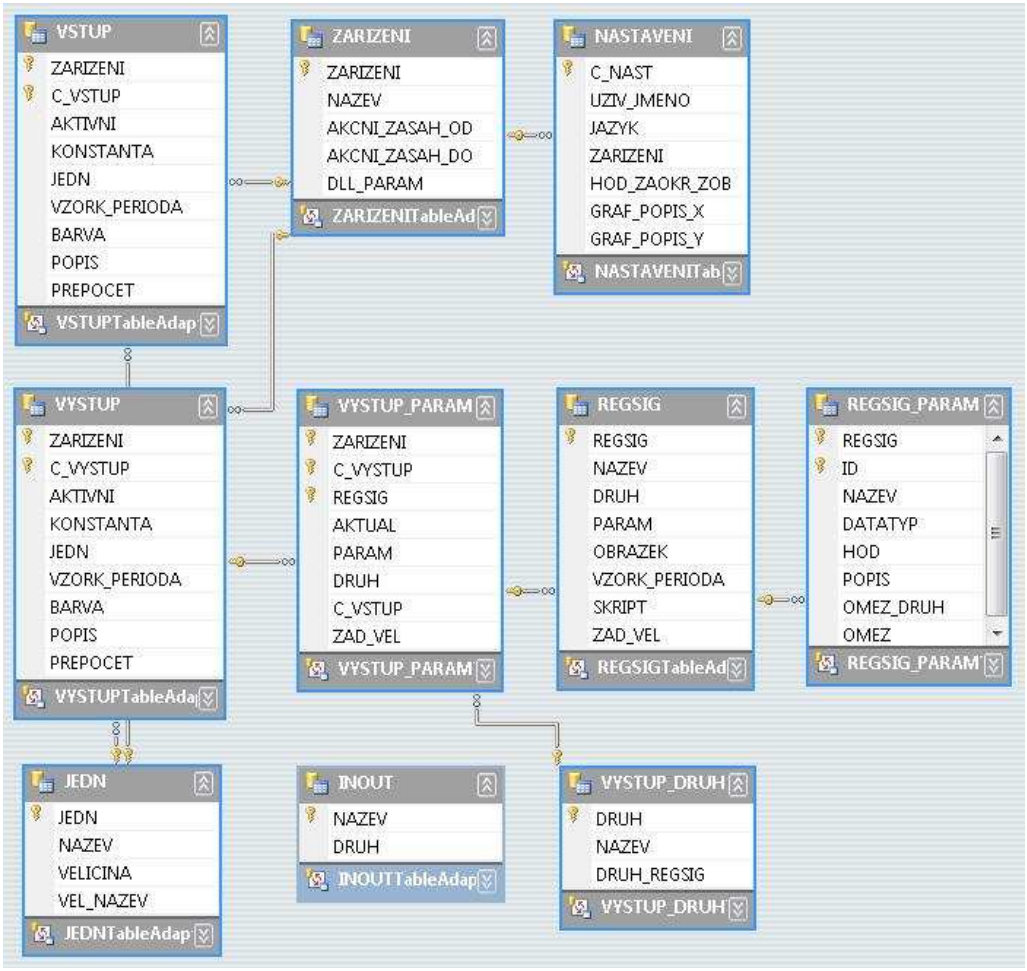
- plná podpora prostředí Visual Studio 2008 a .NET Framework 3.5,
- in-process komponenta,
- jedna databáze je v jednom souboru **.sdf*,
- nenáročná na paměť (max. 2MB pro všechny instance),
- má některé vlastnosti Microsoft SQL Serveru jako jsou transakce, omezení referenční integrity, uzamykání a vícenásobný přístup k jedné databázi,
- podporuje indexování, replikaci vzdálených dat (cashování na lokálním počítači při vzdáleném přístupu) a slučovací replikaci,
- nepodporuje uložené procedury, subtransakce a nativní XML datový typ,
- max. velikost jedné databáze je 4GB,
- databázi lze zabezpečit 128bitovým kódováním a uživatelským heslem,
- jednoduchý export a import databáze - spočívá v kopii a vložení jednoho souboru,
- je zdarma [Microsoft, 2008].

4.2.2 Návrh databáze

Koncepce programu bude zohledňovat přístup více uživatelů a používání více hardwarových zařízení pro vstupní a výstupní komunikaci. Většina parametrů tak bude vázána na používané zařízení. Lze to shrnout tak, že každý uživatel bude mít své vlastní nastavení pro každé používané zařízení. V tabulce 4-1 jsou popsány databázové tabulky včetně krátkého popisu jejich určení. Na obrázku 4-3 je znázorněno schéma vytvořené nástrojem *Dataset Designer* v prostředí Visual Studio 2008. Jsou zde zřejmé jednotlivé sloupce tabulek a označení primárních klíčů. Jsou zde definovány také vazby (*relationships*) mezi tabulkami. [Farana 1999; Farana 2006]

Tabulka 4-1 – Přehled databázových tabulek a jejich určení

Tabulka	Určení
ZARIZENI	Hardwarová zařízení a jejich parametry
REGSIG	Regulátory / signály a jejich parametry
REGSIG_PARAM	Jednotlivé parametry regulátorů / signálů
VSTUP	Vstupy zařízení a jejich parametry
VYSTUP	Výstupy zařízení a jejich parametry
VYSTUP_DRUH	Druhy výstupů – regulátor, signál apod.
VYSTUP_PARAM	Přiřazení regulátorů a signálů k výstupu
NASTAVENI	Parametry programu a objektů
JEDN	Veličiny a jejich jednotky
INOUT (*view*)	Pohled (*view*) pro seznam vstupů a výstupů



Obrázek 4-3 – Databázový model v nástroji Dataset Designer prostředí Visual Studio 2008

5 Nástroje pro realizaci programové aplikace

V této kapitole si popíšeme nástroje, které budou použity k realizaci výsledné aplikace. Zaměříme se na vývojové prostředí, programátorské postupy a nástroje, které budou při vývoji používány. Na závěr zvolíme vhodný nástroj pro grafickou prezentaci dat.

5.1 Visual Studio 2008 a .NET Framework 3.5

Platforma .NET byla oficiálně představena firmou Microsoft v roce 2000 jako klíčový produkt, jehož rozvoj a propagace je součástí dlouhodobé strategie firmy. Jde o novou generaci systému vývoje aplikací pro operační systémy Windows založených na řízeném běhovém prostředí s rozsáhlou sadou základních tříd nazvaném .NET Framework. [Kačmář 2001; Albahari 2002; Prosise 2003; Puš 2006; Hilyard & Teilhet 2007; Troelsen 2007]

Hlavními důvody více než čtyřletého vývoje, jehož výsledkem je .NET, byly:

- nekompatibilita jednotlivých programovacích jazyků a s tím související obtížná spolupráce mezi programy napsanými v odlišných jazycích (např. C++ a Visual Basic),
- vysoká chybovost aplikací (chyby v práci s pamětí, neplatné konverze datových typů),
- problémy s verzemi knihoven (obtížná práce s provozem více verzí knihoven),
- zastaralý a nepřehledný způsob vývoje dosavadních webových aplikací.

Všechny tyto problémy efektivně řeší platforma .NET použitím již zmíněného řízeného běhového prostředí, systémem *assemblies*, což jsou základní stavební prvky aplikací, a novou technologií ASP.NET pro vývoj webových aplikací.

5.1.1 Princip běhového prostředí

Většina dnešních aplikací vytvořených například v jazyce C++, Visual Basic nebo Delphi jsou zkompilovány přímo pro danou platformu, nejčastěji je to pro platformu Win32 operačních systémů Windows. Zdrojový kód je tedy kompilací převeden do strojového kódu počítače. To přináší velmi dobrou rychlost běhu výsledné aplikace. Z toho ale plynou i některé nevýhody – nepřenositelnost mezi jednotlivými platformami popř.

verzemi operačních systémů a časté chyby v přístupech do operační paměti. Princip řízených běhových prostředí u platformy .NET na to jde trochu jinak a přidává k převodu zdrojového kódu do kódu strojového ještě jednu vrstvu. Tuto vrstvu představuje mezikód, do kterého jsou zdrojové kódy zkompilovány, a tento mezikód je běhovým prostředím na cílové platformě (Windows, Linux) převeden do strojového kódu. Tento převod je na cílové platformě realizován vždy při spouštění konkrétní aplikace. Mínusem tohoto překladu je vyšší náročnost na výkon uživatelského počítače, a z tohoto důvodu se tento způsob nepoužívá pro vývoj výpočetně náročných aplikací (např. počítačových her). S jeho častým použitím se naopak můžeme setkat spíše u obchodních aplikací, které nejsou tak náročné na výpočetní výkon a rychlost běhu daných aplikací je naprosto vyhovující. U spousty úloh (přístup k databázi, souborům atd.) uživatel snížení rychlosti aplikace ani nepocítí. U těchto běhových prostředí je důležité, že při spuštění aplikace nedochází k překladu celé aplikace najednou, ale používá se JIT (*Just-in-Time*) kompilace. JIT kompilace znamená, že do strojového kódu je převedena pouze potřebná část mezikódu a při opětovném použití této (již přeložené) části se spouští její zkompilovaná forma, což se příznivě projeví na rychlosti, která si již nezadá s během neřízeného programu.

5.1.2 Klíčové vlastnosti

Již zmíněný mezikód se v této platformě nazývá MSIL, tedy *Microsoft Intermediate Language*. Tento jazyk relativních adres je spouštěn klíčovou součástí .NET Framework pojmenovanou CLR (*Common Language Runtime* neboli společné běhové prostředí). V prostředí CLR existuje tzv. *Garbage Collector*, který programátorům velmi usnadňuje práci s operační pamětí. Jedná se o sadu složitých algoritmů pro uvolňování nepotřebných programových objektů z paměti, díky nimž se již vývojáři nemusejí starat o přiřazování nebo uvolňování operační paměti a odpadá tak riziko již zmíněné nekorektní práce s ní, která ve většině situací končí pádem aplikace.

Velmi důležitou vlastností platformy je CLS – *Common Language Specification* (společná jazyková specifikace). S ní souvisí CTS neboli *Common Type System* (společný typový systém). Výsledkem použití CLS a CTS je rovnocennost programovacích jazyků. Pro vývoj .NET aplikací je tedy možné použít jeden z několika programovacích jazyků vyšší úrovně. Může se jednat například o C#, nový jazyk vyvinutý pro .NET, Visual Basic .NET, nová generace oblíbeného jazyku Visual Basic, J#, což je jazyk se syntaxí

rozšířeného jazyka Java, managed C++, kde slovo *managed* označuje možnost psát řízený kód pro .NET.

S tím souvisí i další výhoda této platformy – výrobcům třetích stran nic nebrání ve vývoji dalších jazyků. Jediné, co stačí, je, aby tento nový jazyk měl kompilátor se schopností kompilovat zdrojové kódy do jazyka MSIL, to znamená splnit specifikaci CLS danou společností Microsoft.

5.1.3 Typy aplikací

Platforma Microsoft .NET vývojářům nabízí široké možnosti. Nejlépe je začít u klasických konzolových aplikací, které pro vstup a výstup používají příkazový řádek. Zajímavější však jsou aplikace s využitím knihoven *Windows Forms*, interně využívající Microsoft Win32 API. Výsledkem jejich použití jsou známé formulářové aplikace pro Windows. Možné je také vytvořit aplikaci běžící jako proces na pozadí systému – službu Windows.

Dalším odvětvím jsou webové aplikace nahrazující zastaralé ASP 2.0, a to jejich nová generace označovaná jako ASP.NET, která nabízí široké možnosti pro tvorbu dynamických webových projektů. Zajímavým typem aplikací jsou takzvané webové služby, které umožňují pomocí http protokolu na vzdáleném serveru volat metody. Dále je možno tvořit knihovny tříd, bez které by vyspělá platforma Microsoft .NET neměla velký smysl.

5.2 Programovací jazyk C#

C# (čti [:sí šárp:]) je nově vyvinutý jazyk pro Microsoft .NET. Je navržen pro maximální využití této platformy. Jedná se o objektově orientovaný jazyk vycházející z programovacích jazyků Java a C++. Stejně jako tyto jazyky je i C# *case-sensitive*, což znamená, že významově odlišuje velká a malá písmena ve výrazech (winctrl a Winctrl jsou brány jako dva rozdílné pojmy). V tomto jazyce je realizováno 80% základních knihoven prostředí .NET Framework. I přes to, že je koncipován hlavně pro psaní řízeného kódu, na jehož užití je platforma .NET postavena, lze jej v případě potřeby využít i pro tvorbu kódu neřízeného. Použití neřízeného kódu znamená, že běhové prostředí CLR neověřuje, zda je napsaný kód bezpečný (například se neověřuje jinak vyžadovaná typová bezpečnost).

5.2.1 Vlastnosti jazyka

Vlastností jazyka C#, které lze použít při tvorbě aplikací:

- **Třídy** – základní stavební prvek při tvorbě objektově orientovaných aplikací obsahující akce (metody) a atributy.
- **Struktury** – lze je chápat jako zjednodušené třídy, jejich užitím jsou nejčastěji popisovány vlastní datové struktury.
- **Výčtové typy -** pojetí je oproti Javě značně zjednodušené. Výčet je pouze množina předem definovaných hodnot bez možnosti definovat metody, atributy atd.
- **Vlastnosti** – někdy označované jako chytré proměnné.
- **Pole** a jejich vylepšená verze nazývaná indexery.
- **Zástupci** – typově bezpečné ukazatele na funkce.
- **Události** – druh zástupců sloužící ke zpracování asynchronních operací.

Základním požadavkem pro běh .NET aplikací je samozřejmě běhové prostředí .NET Framework. Pro vývoj je potřeba stáhnout instalátor *.NET Framework SDK* (*Software Development Kit*), který mimo jiné obsahuje obsáhlou dokumentaci k základním třídám prostředí .NET Framework.

5.2.2 Vývojová prostředí

Pro vývoj .NET aplikací je dispozici několik vývojových prostředí. Kód aplikace lze psát i v poznámkovém bloku a následně jej kompilovat v prostředí příkazového řádku. Nejvhodnějším vývojovým prostředím je pravděpodobně Visual Studio .NET respektive Visual C#, což je jeho součást. Bohužel toto vývojové prostředí není zdarma. Za účelem studia lze použít Visual Studio Express, které je zdarma ke stažení na stránkách společnosti Microsoft nebo také *C# Builder* od firmy Borland, který je ve své verzi *Personal* volně ke stažení. Tuto verzi není povoleno používat pro vývoj komerčních aplikací. Pro vývoj komerčních aplikací existuje open source vývojové prostředí SharpDevelop. [Puš 2006]

Obrázek 5-1 – Vývojová prostředí - Visual Studio Team System, Express Edition a open source SharpDevelop

5.3 Tvorba komponent ve Visual Studio 2008 a C#

Aplikace vytvořené technologií .NET se skládají z komponent. Všechny objekty obsahují vlastnosti, metody a události, které jsou základem objektově orientovaného programování (OOP). Velmi vhodné je při návrhu a realizaci projektu oddělit funkční část programu od prezentační. To znamená, že je program rozdělen na několik dílčích částí, které spolu komunikují podobně jako by byly v jednom programu, existují však samostatně a mohou být využívány různými vzájemně nezávislými klientskými aplikacemi. Lze to např. vysvětlit tak, že existuje služba, která řeší veškeré výpočetní operace a poskytuje své metody pro jiné aplikace. Prezentační aplikace volá metody služby, posílá vstupní data, zpracovává a zobrazuje data výstupní. Takto je možné upravovat jednotlivé dílčí aplikace, které při zachování konvencí (ty musí být stanoveny na začátku projektu, nejlépe již při jeho modelování) pracují se stále stejným typizovaným formátem dat. Prezentační část aplikace pak může jakýmsi unifikovaným způsobem komunikovat s různými komponentami, které jsou např. úzce vázány na určitý hardware.

Komponenta je zvláštním typem výkonného souboru vytvořeného z vývojového prostředí technologie .NET. Po kompilaci komponenty se na ni obvykle v aplikaci, která ji bude využívat, přidá odkaz (reference). Velmi často komponenty běží na webovém serveru a poskytují data a služby webovým servisům. U desktopových aplikací to funguje obdobně, avšak ne v takovém rozsahu.

Samotná tvorba komponenty není složitější než vytvoření běžné třídy, do které se přidají potřebné vlastnosti, metody a události a na kterou se vytvoří odkazy v klientských aplikacích. .NET komponenta je vlastně předkompilovaná třída s příponou **.dll* (*dynamically-linked library*). Při spuštění je .NET komponenta zavolána a načtena do paměti, kde je připravena k použití pro klientské aplikace. Takovéto komponenty jsou většinou vytvářeny a testovány jako samostatné projekty a nemusí být nutně součástí nějakého projektu (nemusí být volány z externí klientské aplikace). Po kompilaci do knihovny **.dll* mohou být komponenty přidány do klientských aplikací. Každá komponenta může obsahovat libovolný počet tříd. Např. komponenta pro získávání dat z hardwarového zařízení může mít zvlášť třídy pro komunikaci, převod formátů výstupních dat nebo záznam dat. Nebo může existovat komponenta s pouze jednou třídou např. pro zpracování chyb v aplikaci. [Groh 2002]

5.3.1 Komponenty v paměti počítače

Všechny aplikace platformy Microsoft Windows běží v paměti počítače a používají jeho prostředky jako je diskový prostor, síťové služby a grafická rozhraní. Moderní systémy Windows jsou již více-úlohové (*multi-tasking*) operační systémy, což znamená, že více aplikací najednou sdílí paměť počítače a další jeho možnosti. Systém Windows řídí chod aplikací, dynamicky jim přiděluje paměť prostřednictvím procesů. Paměť je obsazena aplikací a jejími daty pouze po dobu časového kvanta, doby, která je přidělena jednotlivým vláknům procesu. Vlákna jsou dílčí výpočetní úkoly procesu. Paměť je přidělována dynamicky podle potřeby aplikací a jejich priorit. Plánovač systému Windows přiděluje paměť dle požadavků aplikací a sleduje, zda již vypršel čas vlákna. Pak automaticky předává uvolněnou paměť dalším vláknům. V některých případech mohou být procesy sdíleny mezi aplikacemi a to v případě, že je potřeba mezi aplikacemi sdílet data, služby apod. Jindy je paměť osazena pouze procesem jedné samostatné **.exe* aplikace.

Obrázek 5-2 - Ikona komponenty dll v operačním systému Windows Vista

Mnoho aplikací jako je např. Microsoft Word nebo Excel podporují širokou škálu funkcí. Než by měly být aplikace ve formě jednoho rozměrného souboru, bylo vhodnější je rozdělit na dílčí funkční celky, které je pak možno využít v obou programech. Hlavní soubory programů mají příponu **.exe*, zatímco jednotlivé funkční celky jsou v souborech s příponou **.dll*.

Pokud hlavní program potřebuje využít funkčnost obsaženou v **.dll* knihovně, přidá ji do procesu hlavní aplikace. Systém Windows pracuje s hlavním programem a vykonávaných kódem **.dll* knihovny v rámci jednoho procesu. Volání komponenty **.dll*, která je součástí procesu jiné aplikace, se nazývá *in-process* (v procesu). Existují i situace, kdy nejsou komponenty volány jako *in-process*. To je např. v situaci, kdy z programu chceme tisknout dokument. Aplikace vyžádá od systému Windows příslušný ovladač a ten načte do paměti. Pokud stejný ovladač potřebuje jiná aplikace, Windows nahraje ovladač jako *out-of-process* (mimo proces) službu, v tom případě může ovladač využívat více

aplikací najednou. Z toho vyplývá také skutečnost, že systém Windows nemůže načíst do paměti více kopii jednoho ovladače.

In-process komponenty

.NET komponenty ve formátu **.dll* pracují v rámci procesu hostitelské aplikace, sdílejí paměť a požadavky na procesor s hostitelskou aplikací. Při spuštění je komponenta načtena z disku a je přidána k procesu hostitelské aplikace. Protože nejsou volány žádné externí procedury pro zprostředkování komunikace mezi komponentou a hostitelskou aplikací, probíhá nastavování a zjišťování hodnot vlastností, volání metod a reakce na události vyvolané v komponentě velmi rychle.

Out-of-process komponenty

Tato alternativní architektura zahrnuje serverové komponenty, které běží samostatně a nezávisle na procesech aplikací, které komponenty využívají. Serverové komponenty mají většinou příponu **.exe*. Jakmile systém Windows načte *out-of-process* komponentu, je pro ni vytvořen separátní proces, který je řízen dle požadavků komponenty nezávisle na klientské aplikaci. Windows zprostředkovává komunikaci mezi serverovou a klientskou aplikací prostřednictvím zpráv.

5.3.2 Výhody .NET komponent

Oproti starším vývojovým prostředím je výhoda především v tom, že

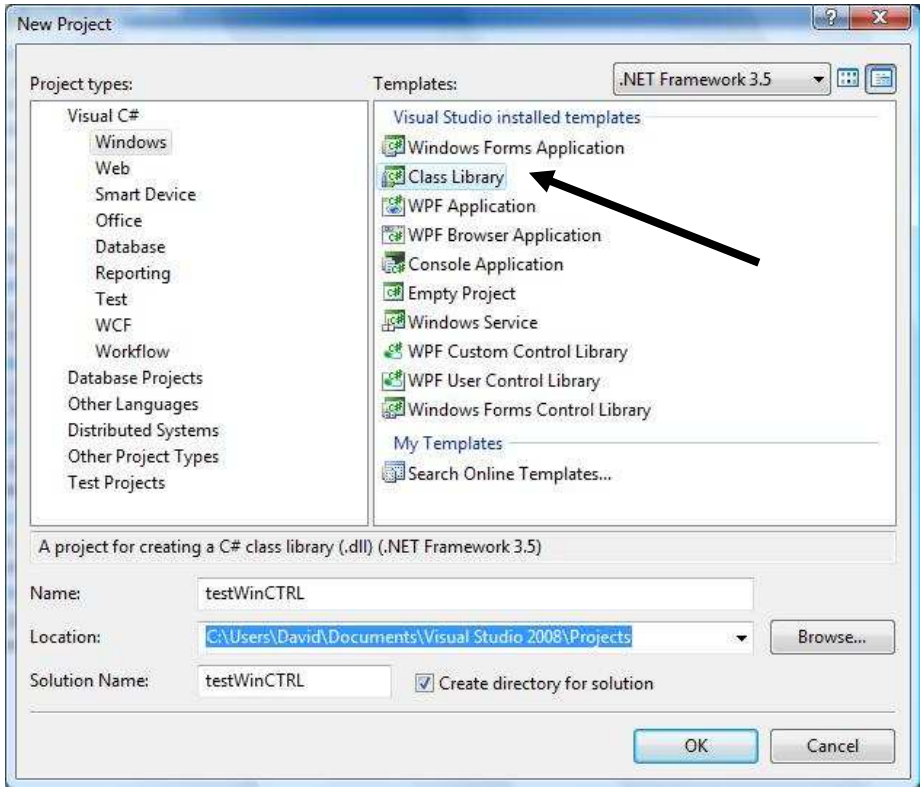
- *In-process* komponenty běží rychleji, protože nejsou volány žádné vzdálené procedury. Klientské aplikace mají přímý přístup k prostředkům komponenty bez nutnosti využívat komunikačního dialogu.
- *Out-of-process* komponenty se mohou v některých případech osvědčit lépe než *in-process* komponenty, neboť v případě, že *out-of-process* aplikace havaruje, nezpůsobí zároveň havárii klientské aplikace. U *in-process* aplikací znamená havárie komponenty zpravidla také havárii klientské aplikace.
- *Out-of-process* komponenty jsou jednodušeji sdíleny řadou klientských aplikací. Pokud je komponenta zkompilovaná pro možnost sdílení, načte ji systém Windows jako jednu kopii *out-of-process* komponenty a umožní ji sdílet libovolným počtem

klientů. Samozřejmě může dojít ke snížení výkonu v případě, že je ve stejném čase komponenta volána velkým počtem klientských aplikací.

5.3.3 Postup při tvorbě komponent

Tato kapitola se bude zabývat pouze tvorbou komponent, bude předpokládat základní znalosti programování a architektury Visual Studia .NET.

Pro vytvoření komponenty je nutno nejprve založit projekt v prostředí Visual Studio. Při specifikaci projektu je nutno místo klasické *Windows Application* zvolit *Class Library*, tedy knihovnu tříd. Náš zkušební projekt nazveme *testWinCTRL*.



Obrázek 5-3 - Založení projektu pro komponentu

V založeném projektu již pracujeme stejným způsobem jako u běžné aplikace, vkládáme tedy jednotlivé třídy, metody atd. Pro lepší představu se zaměříme na třídu *testWinCTRL* a její dvě veřejné metody pro zápis a čtení. Každá metoda používá lokální metody, ty nejsou přístupné přímo přes komponentu.

```
public void zapis(string text,out string chyba)
{
    pripoj();
    // zapise string
```

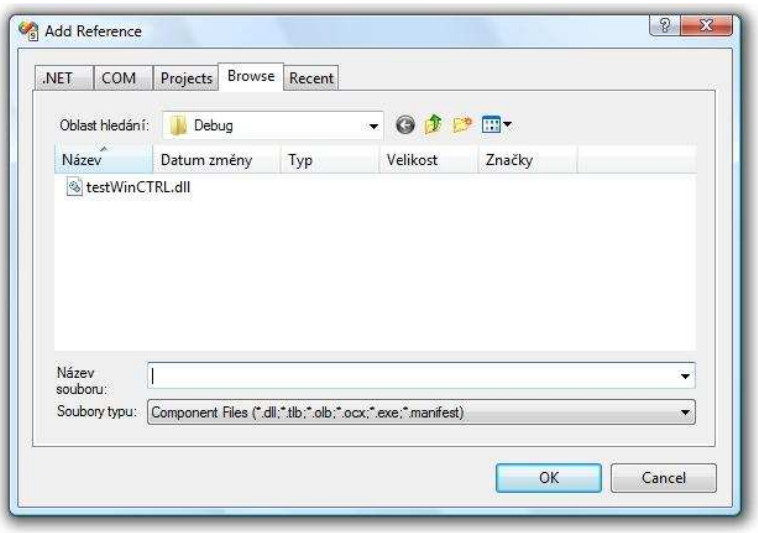
```
        try
        {
            //prevede ascii na byte vcetne LF a CR
            byte[] data = System.Text.ASCIIEncoding.Default.GetBytes("@01 " +
          text + "\r\n");
            //zapise
            port.Write(data, 0, data.Length);
            chyba = "";
        }
        catch
        {
            chyba="Došlo k chybě při zápisu na sériový port!";
        }
}
public void cteni(out string text,out string chyba)
{
        pripoj();

        try
        {
            text = uprav_hod(port.ReadTo("\r\n"));
            chyba = "";
        }
        catch
        {
            text="";
            chyba = "Došlo k chybě při čtení ze sériového portu!";
        }
}
```

Nyní můžeme provést překlad projektu a následně v adresáři *Debug* najít soubor s názvem *testWinCTRL.dll*. Toto je již vyexportovaná komponenta, kterou je možno ihned použít v dalších aplikacích.

5.3.4 Použití komponent v klientských aplikacích

Pro použití komponenty v jiné aplikaci je nejdříve nutno přidat referenci na komponentu.



Obrázek 5-4 - Přidání reference na komponentu

V programovém kódu klientské aplikace musíme také přidat jmenný prostor komponenty.

```
using testWinCTRL;
```

A dále již postupujeme standardním způsobem tzn. založením instance třídy. Pak pracujeme s komponentou jako s lokální třídou.

```
//instance komponenty
WinCTRL winCTRL = new WinCTRL();
```

Programový kód prezentační testovací aplikace pro čtení a zápis dat je následující.

```
private void button1_Click(object sender, EventArgs e)
{
    string zprava;
    string text;
    string hod;
    int pocet_hod=0;

    //instance pro nahodna cisla
    Random random = new Random();
    do
    {
        //zapise na analog vystup 1 nahodnou hodnotu od 200 do 1500
        winCTRL.zapis("N0 " + random.Next(200, 1500), out zprava);
        //precte hodnotu
        winCTRL.cteni(out text, out zprava);
        //zmeri analog vstup 1
        //zapise dotaz na hodnotu
     //dokud se meni hodnota, kresli graf
        do
        {
            hod = text;
            winCTRL.zapis("A0", out zprava);
            //precte hodnotu
            winCTRL.cteni(out text, out zprava);
            //zapise hodnotu do grafu
            chart1.Series["Series1"].Points.AddY(text);
            pocet_hod++;
        }
        while (hod != text);
        label1.Text = pocet_hod.ToString();
    }
    while (pocet_hod < 1000);
}
```

Procedura pracuje tak, že posílá do jednotky příkazy k nastavení analogového výstupu 1 a ihned pak zjišťuje stav analogového vstupu 1. Pro tento testovací projekt je použito zapojení s kondenzátorem a rezistorem. Zápisem na výstup dochází k nabíjení a poté k plynulému vybíjení kondenzátoru a toto měnící se napětí je měřeno. Protože se napětí velmi rychle ustaluje (v závislosti na původní a následující hodnotě), je procedura ošetřena

tak, že jakmile jsou dvě po sobě jdoucí hodnoty stejné, vygeneruje jiné napětí a opět proběhne měření. Každá hodnota je zároveň zapisována do grafu.

5.4 Dynamická kompilace ve Visual Studiu 2008

Cílem aplikace WinCoMeT je možnost používat různá hardwarová zařízení. V předchozí kapitole jsme vysvětlili možnosti implementace ovládání hardwarového zařízení pomocí komponent. Na tyto komponenty jsme však odkazovali přímo v programovém kódu. Komponenty, na které existoval odkaz (reference) v době překladu aplikace, bude možno bez omezení používat, bude možno vytvářet instance tříd, volat jejich metody atd. Tyto komponenty resp. jejich soubory **.dll* budou umístěny v adresáři aplikace s ostatními strukturami aplikace.

Nyní musíme vyřešit možnost přidání nové komponenty, takové, která neexistuje v době překladu aplikace, na kterou tedy nemůžeme vytvořit před kompilací odkaz. Budeme muset využít možnost dynamické kompilace. Jedná se o třídy jmenného prostoru *System.Reflection*, na který se v programovém kódu musíme odkázat.

```
using System.Reflection;
```

Pak již můžeme využít třídy *Assembly*, která poskytuje metody pro načtení komponenty do aplikace a možnost jejího plnohodnotného využití. Máme-li např. komponentu ve formě **.dll* souboru a chceme ji využít v programovém kódu, pomocí třídy *Assembly* ji načteme následujícím způsobem.

```
string str = Application.StartupPath + "\\Device\\" + device + ".dll";
Assembly asm = Assembly.LoadFile(str);
```

Pomocí proměnné *asm* přistupujeme ke komponentě např. tak, že si do proměnné načteme námi známou třídu a ihned vytvoříme její instanci.

```
Type trida = asm.GetType("CTRLv3");
object instance = Activator.CreateInstance(trida);
```

K vlastnostem tříd a hodnotám vlastností konkrétních instancí přistupujeme takto:

```
FieldInfo vlastnost = trida.GetField("pocetVstupu");
int hodnota = (int)vlastnost.GetValue(instance);
```

Pro volání metod nejdříve vytvoříme proměnnou typu *object*, do které vložíme hodnoty vstupních parametrů, poté budeme volat metodu *InvokeMember*, kdy jako vstupní proměnné uvádíme název metody, potřebné přepínače typu *BindingFlags*, instanci dané třídy a objekt vstupních parametrů.

```
object[] parametr = new object[] { "vstup1" };
double navHodnota = (double)trida.InvokeMember("NazevMetody",
BindingFlags.Default|BindingFlags.InvokeMethod, null, instance,parametr);
```

V tomto případě má metoda návratovou hodnotu typu *double*, je proto nutné provést přetypování jinak obecné návratové hodnoty typu *object*. Neznáme-li třídy a metody poskytované komponentou, můžeme je zjistit pomocí metod, které vrací pole (*array*) všech tříd komponenty a všechny metody příslušné třídy.

```
Type[] tridy = asm.GetTypes();
MethodInfo[] metody = trida.GetMethods();
```

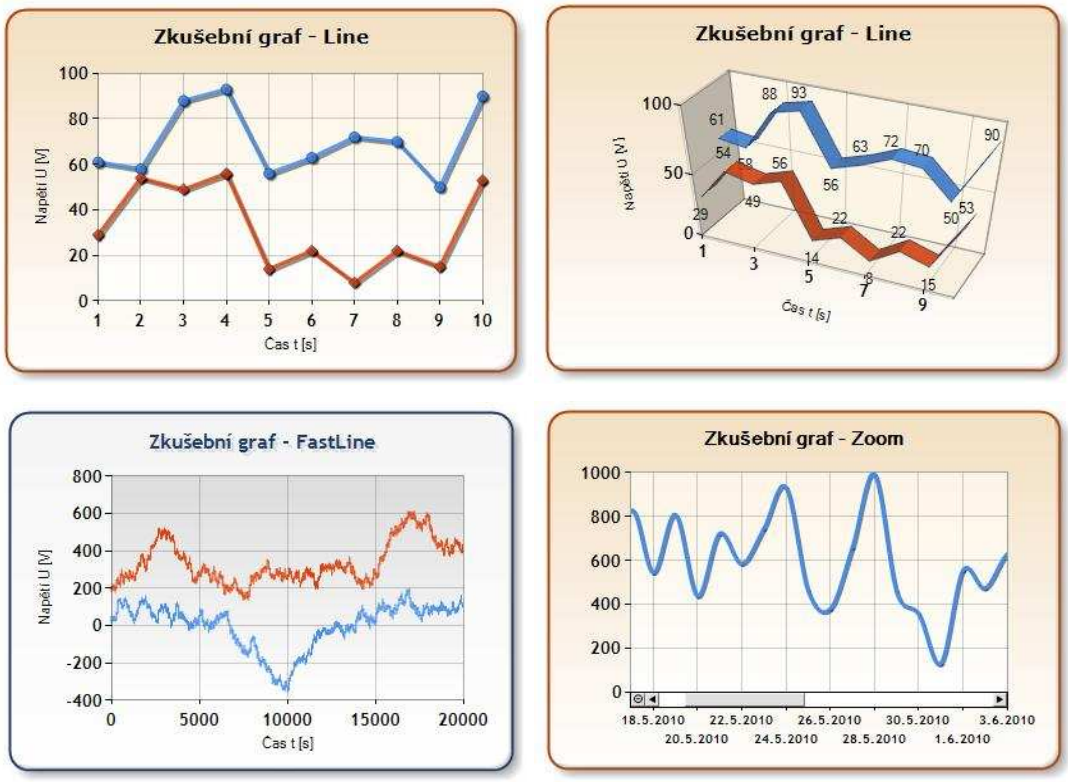
5.5 Prezentace měřených dat

Pro vykreslování změřených dat do grafu byla zvolena komponenta Microsoft Chart Control for .NET Framework. Jedná se o komponentu, která je určena přímo pro vývojové prostředí Visual Studio 2008 a pro .NET Framework 3.5. Tuto komponentu je možno zařadit do nástrojové lišty pro formulářové aplikace a je možno ji standardním způsobem používat.

5.5.1 Microsoft Chart Control

Microsoft Chart Control for .NET Framework poskytuje vývojářům softwarových aplikací možnosti vizualizace dat. Umožňuje vytvářet jednoduché, intuitivní a vizuálně působivé grafy pro široké spektrum použití, pro komplexní statistiky či finanční analýzu. Jde o plně ovladatelnou komponentu prostředí .NET Framework navrženou výhradně pro vývojové prostředí Microsoft Visual Studio 2008.

Tuto komponentu můžeme použít jak ve formulářích desktopových aplikací (*Windows Forms*) a také ve webovém prostředí (ASP.NET). [Microsoft 2008; Gorev 2008]



Obrázek 5-5 – Ukázka grafů komponenty Microsoft Chart Control

Mezi klíčové vlastnosti Microsoft Chart Control patří:

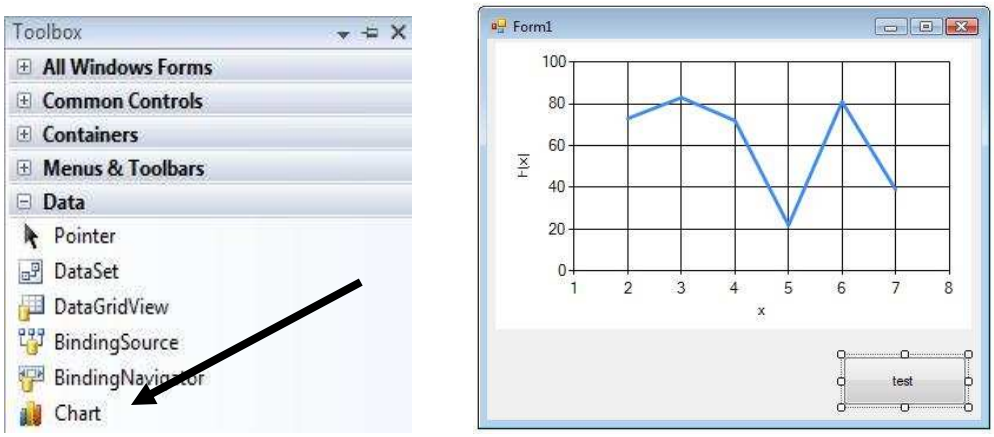
- Plná podpora prostředí Visual studio 2008
- 25 typů grafů
- Podpora 3D zobrazení u většiny grafů
- 3D uživatelská nastavení – perspektiva, osvětlení, rotace, šablony, anti-aliasing, průhlednost, stínování a další
- Neomezený počet oblastí grafů
- Automatické a manuální rozložení grafu a uspořádání objektů
- Automatická a manuální změna měřítka grafu
- Možnost použití logaritmické stupnice
- Plně uživatelsky definovatelná legenda
- Inteligentní umísťování popisků dat (bodů, datových řad)
- Anotace grafů a tooltip
- Zvětšování částí grafů a procházení zvětšených částí pomocí posuvníků
- Propojení grafů s objekty se zdrojovými daty
- Export dat

- Možnost uložení veškerých dat objektu do binárních nebo XML dat
- Neomezený počet datových řad a datových bodů
- Podpora datumových, časových, měnových a dalších jednotek
- Více než 50 druhů finančních a statistických výpočtů pro datovou analýzu
- Podpora editace zobrazovaných dat v reálném čase
- Podpora událostí před zobrazením a po zobrazení grafu

5.5.2 Implementace pro použití v prostředí Visual Studio

Základní požadavky pro instalaci je již nainstalované prostředí Visual Studio 2008 včetně platformy .NET Framework 3.5, dále pak opravný balík prostředí *Visual Studio 2008 Service Pack 1 (SP1)* a opravný balík *Microsoft .NET Framework version 3.5 SP1*.

1. Nejprve je potřeba stáhnout instalační balíček *Chart Control for .NET Framework* z webových stránek společnosti Microsoft.
2. Nainstalujeme knihovny grafů spuštěním souboru *MSChart.exe*
3. Nainstalujeme doplněk pro prostředí Visual Studio 2008 spuštěním souboru *MSChart_VisualStudioAddOn.exe*
4. Po dokončení instalace spustíme Visual Studio 2008 a otevřeme projekt.
5. V oblasti nástrojové lišty (*toolbox*) najdeme v sekci *Data* položku *Chart* (obrázek 5-6)



Obrázek 5-6 – Použití Microsoft Chart Control ve formulářích

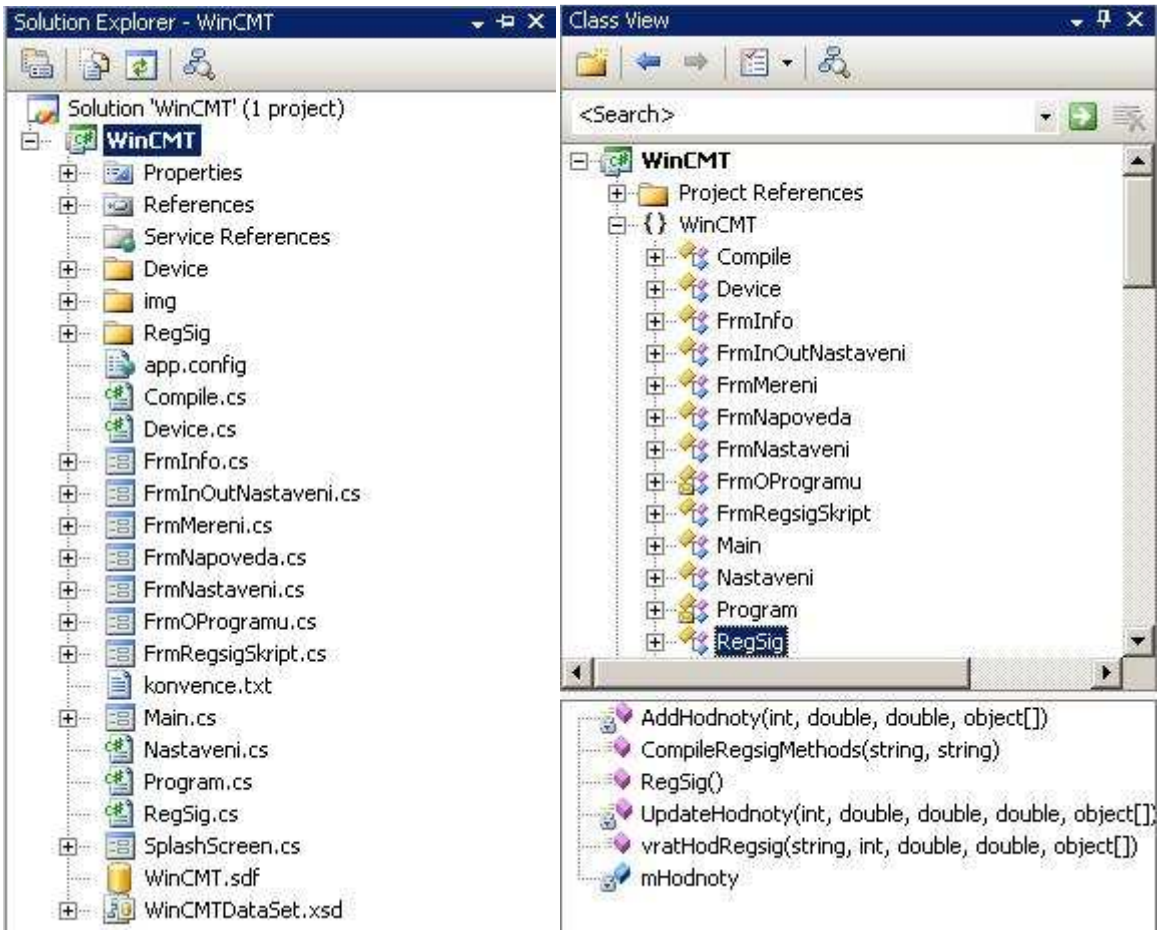
Jednotlivé parametry grafu jsou plně editovatelné a to i za běhu programu. Lze nastavit parametry pro titulek grafu, popis os, pomocné linky, legendu a celou řadu dalších vlastností grafu.

6 Softwarová aplikace WinCoMeT

Nyní přejdeme k vlastní programové aplikaci. Popíšeme zde řešení jednotlivých částí programu s uvedením významných částí programového kódu. Jednotlivé části programu budou popisovány tak, jak byly chronologicky vyvíjeny. [SSW Consulting Services 2008; National instruments 2009]

6.1 Základní informace

Aplikace WinCoMeT byla vytvořena jako řešení a projekt v prostředí Visual Studio 2008. Projekt je zobrazen na obr. 6-1. Skládá se z 8 formulářů a 5 tříd.



Obrázek 6-1 - Přehled součástí projektu ve Visual Studiu 2008 a seznam tříd s výpisem metod

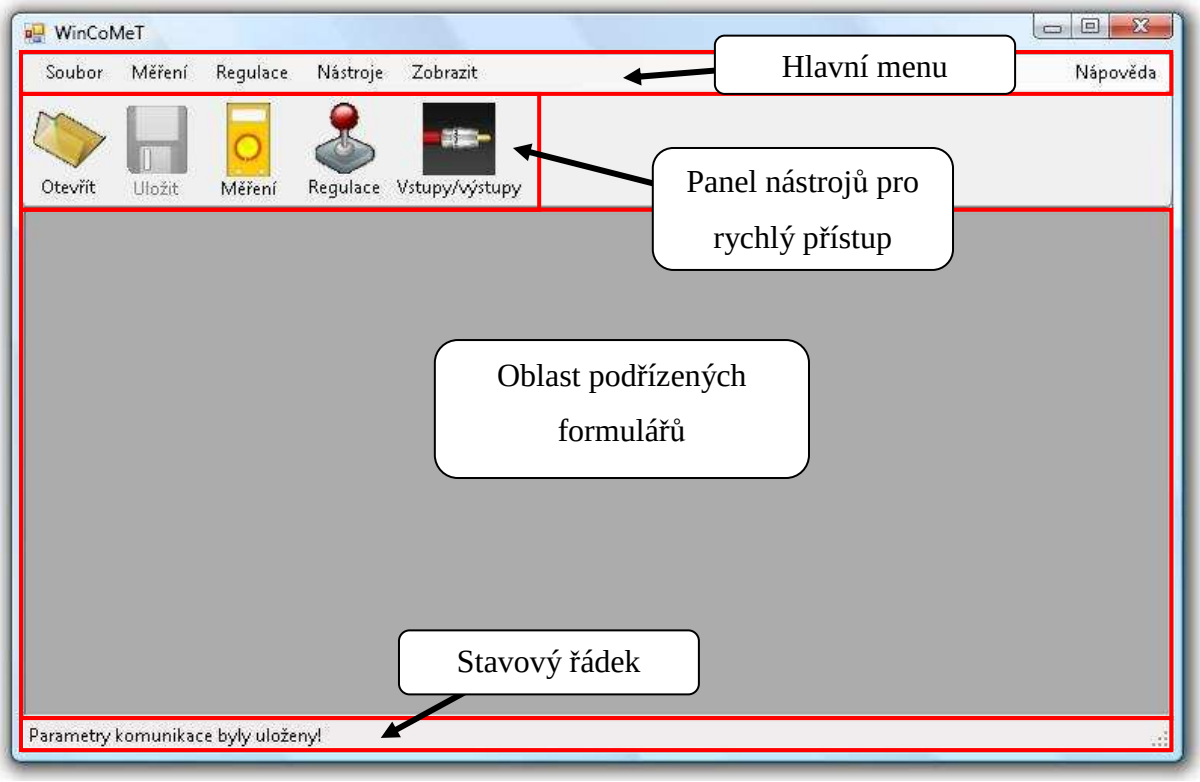
Program je řešen jako MDI aplikace (*Multiple Document Interface*), je zde tedy jeden hlavní formulář *FrmMain.cs*. Všechny ostatní formuláře jsou podřízené, jsou volány z hlavního formuláře. Ze všech podřízených formulářů je možno přistupovat k objektům nadřízeného formuláře a jeho prostřednictvím také k ostatním podřízeným formulářům.

Během spouštění aplikace se objeví tzv. *splashscreen* (speciálně upravený formulář), na kterém je umístěno logo a název programu, aktuální verze a informace o programu. Během zobrazení tohoto formuláře jsou načítány potřebné knihovny programu, dochází k dynamické kompilaci právě zvoleného (nebo implicitního) zařízení, dále pak k dynamické kompilaci metod určených pro výpočet výstupních hodnot regulátorů a generovaných signálů a ověřována přítomnost nové verze programu na internetu.

Obrázek 6-2 – Splashscreen a ikona aplikace

6.2 Ovládání programu

Po spuštění programu je načten hlavní formulář. Ten se skládá ze čtyř hlavních oblastí.



Obrázek 6-3 - Hlavní okno aplikace WinCoMeT

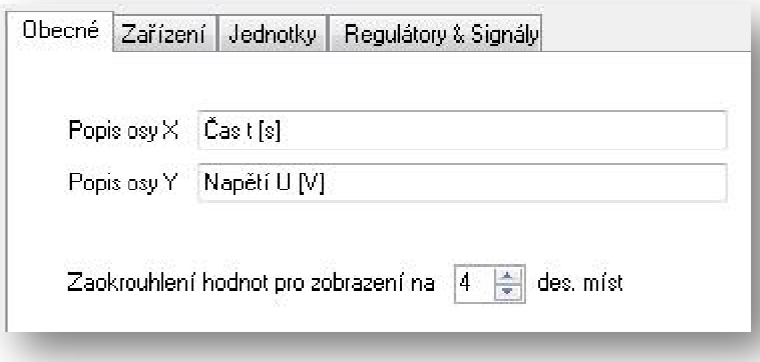
Jsou to

- **Hlavní menu** – ovládání všech funkcí programu. Jedná se o roletové menu, které je účelně rozděleno do pěti sekcí:
  - **Soubor** – ukládání měřených dat a uživatelských parametrů
  - **Měření** – spuštění měření
  - **Regulace** – spuštění regulace
  - **Nástroje** – nastavení programu
  - **Nápověda** – nápověda, aktualizace a informace o programu
- **Panel nástrojů pro rychlý přístup** – urychluje přístup k nejpoužívanějším funkcím
- **Oblast podřízených formulářů** – slouží k zobrazení podřízených formulářů
- **Stavový řádek** – obsahuje informační zprávy programu

6.3 Obecná nastavení programu

Pro nastavení programu slouží dva formuláře. První z nich slouží pro obecná nastavení programu, volbu zařízení, definici jednotek, definici regulátorů a generovaných signálů. Druhý formulář obsahuje nastavení vstupů a výstupů aktuálně zvoleného (nebo implicitně nastaveného) zařízení. Ve formuláři obecného nastavení jsou 4 záložky:

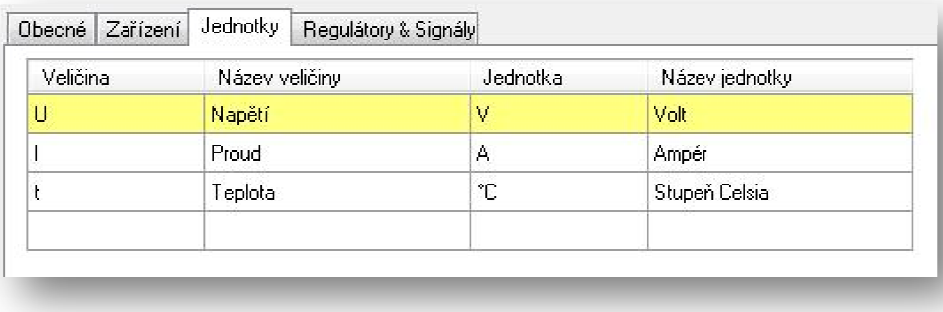
- **Obecné** – pro nastavení popisu grafu a formátu zobrazování měřených hodnot



Obrázek 6-4 – Obecné nastavení aplikace

- **Zařízení** – pro výběr zařízení, konfiguraci parametrů komponenty zařízení, pro nastavení omezení akčního zásahu. Akční zásah lze takto shora i zdola omezit dle vlastností použitého hardwarového zařízení.

- **Jednotky** – pro definici jednotek, které je pak možno přiřadit ke vstupům a výstupům zařízení.



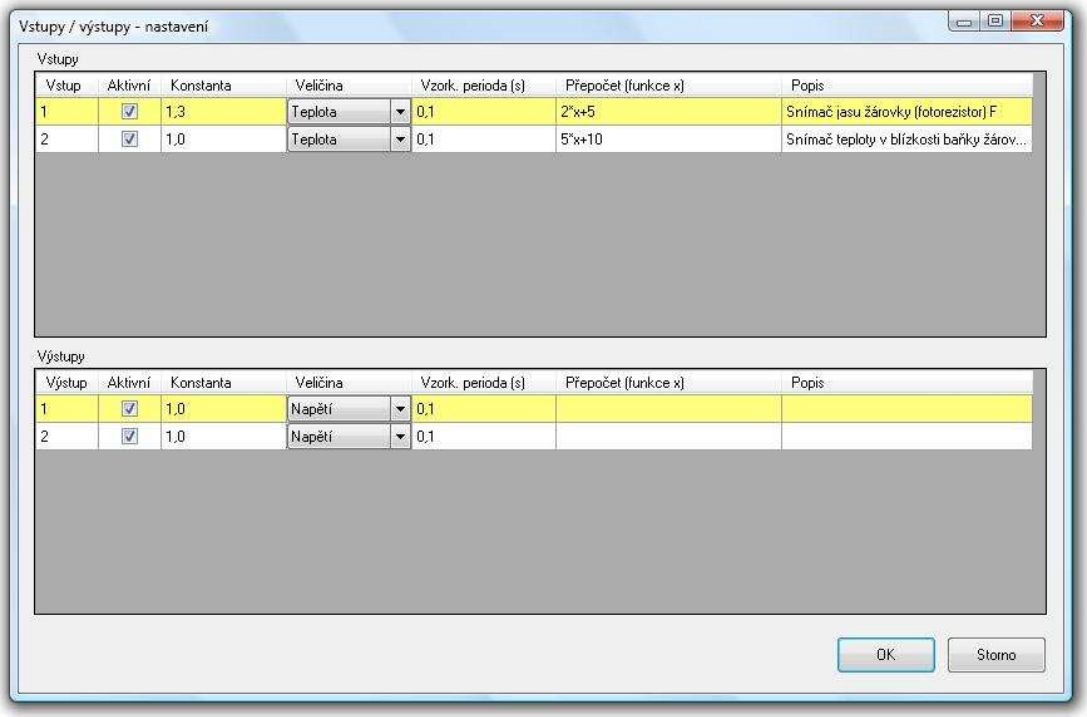
Obrázek 6-5 – Nastavení veličin a jejich jednotek

- **Regulátory / signály** – pro definici parametrů regulátorů a generovaných signálů. Přidáváním a editací nových regulátorů a signálů se zabývá kapitola 6.7.

6.4 Nastavení vstupů a výstupů

Je-li zvoleno zařízení, je možno nastavit parametry jejich vstupů a výstupů. Pro všechny vstupy i výstupy lze nastavit tyto parametry:

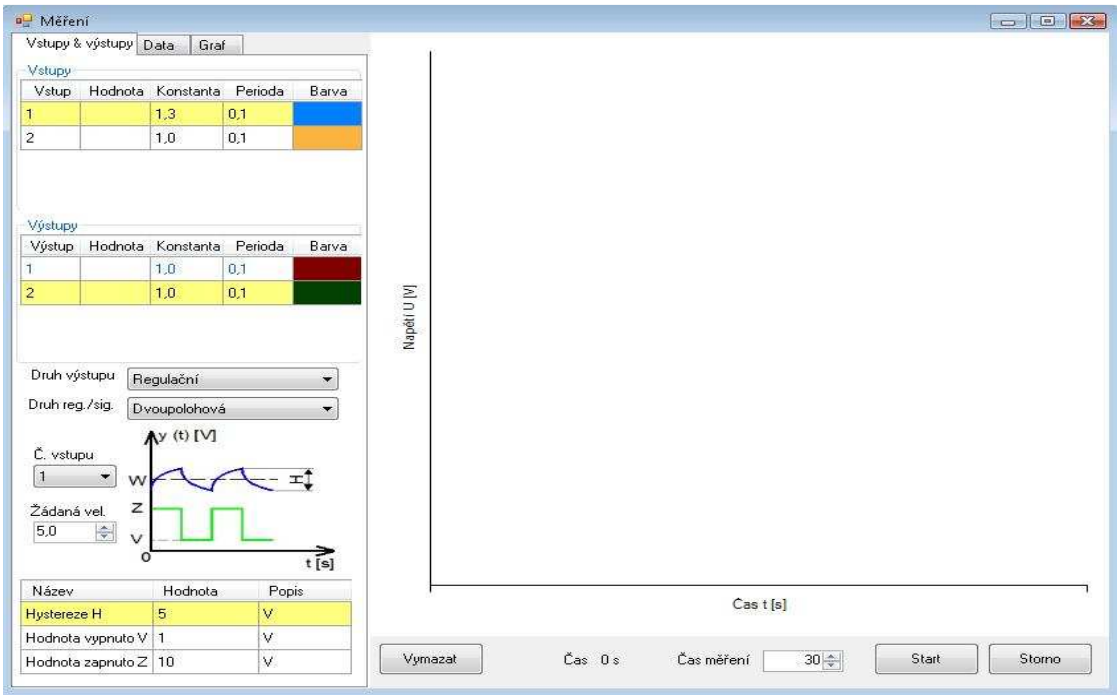
- **Stav aktivní / neaktivní** – znamená, zda se bude daný vstup či výstup používat. Pokud není označený jako aktivní, není zobrazen v dalších formulářích.
- **Konstanta –** číselná konstanta, kterou je změřená hodnota vynásobena.
- **Veličina –** veličina vyjadřující rozměr výsledné změřené resp. vypočtené hodnoty.
- **Vzorkovací perioda –** perioda vzorkování v sekundách – minimální a také implicitní hodnota je 0,1 s.
- **Přepočet –** lze zadat polynom (funkci proměnné x), dle kterého se výsledná hodnota přepočítá. Lze tak při známé charakteristice daného snímače převést změřenou hodnotu na výslednou hodnotu jiného rozměru (např. napětí na teplotu u termistorů). Nemáme-li charakteristiku snímače k dispozici od výrobce, můžeme ji určit pomocí experimentální identifikace.
- **Popis –** popis vstupu / výstupu.



Obrázek 6-6 – Nastavení parametrů vstupů a výstupů

6.5 Měření a regulace

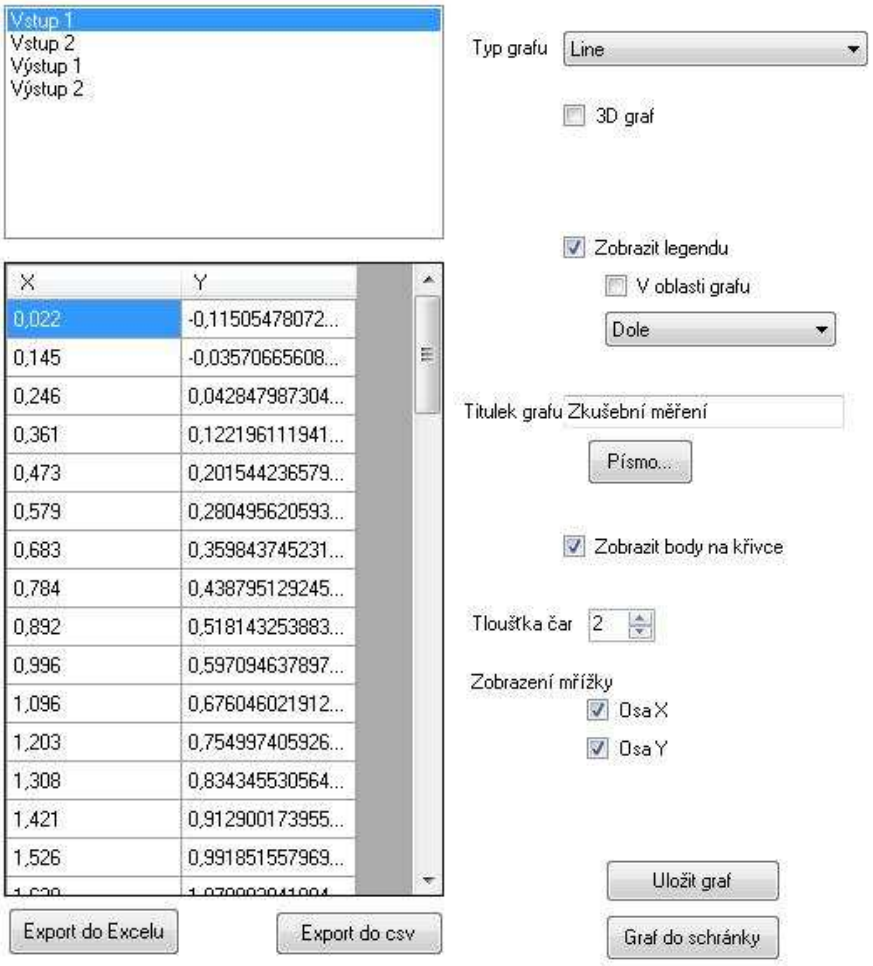
V této kapitole se budeme zabývat výhradně regulací, pro měření slouží stejný formulář i stejný programový kód, avšak funkce související s výstupy a regulací nejsou používány a formulářové prvky jsou skryty.



Obrázek 6-7 – Formulář měření / regulace

Kliknutím na ikonu regulace se spustí maximalizovaný formulář pro měření a regulaci. Jde o podřízený formulář MDI aplikace, zůstává proto k dispozici hlavní menu, stavový řádek a také panel pro rychlé spuštění (samozřejmě závisí na tom, zda je zobrazen panel rychlého spuštění a stavového řádku již před spuštěním regulace).

Pro zobrazení parametrů vstupů a výstupů a také pro parametry regulátorů a signálů slouží standardní prvek prostředí Visual Studio a to je *DataGridView*. Všechny objekty *DataGridView* jsou propojeny s databázovými tabulkami a jejich obsah je načítán při inicializaci formuláře. Všechny parametry je možno měnit i během měření, lze měnit např. žádanou veličinu nebo zesílení regulátoru. Stisknutím tlačítka *Start* se spustí regulace, která trvá dle zadané doby v sekundách, avšak kdykoli ji lze zastavit stisknutím stejného tlačítka, jehož popis se dynamicky mění. Po ukončení regulace se naplní hodnoty v záložce *Data*. Pro vstupy a výstupy je vytvořeno v SQL databázi *view*, je tedy jeden seznam pro všechny datové řady (vstupy i výstupy) a při kliknutí na konkrétní vstup či výstup bude zobrazena dvojice hodnot - čas a změřená (funkční) hodnota (obrázek 6-8a).

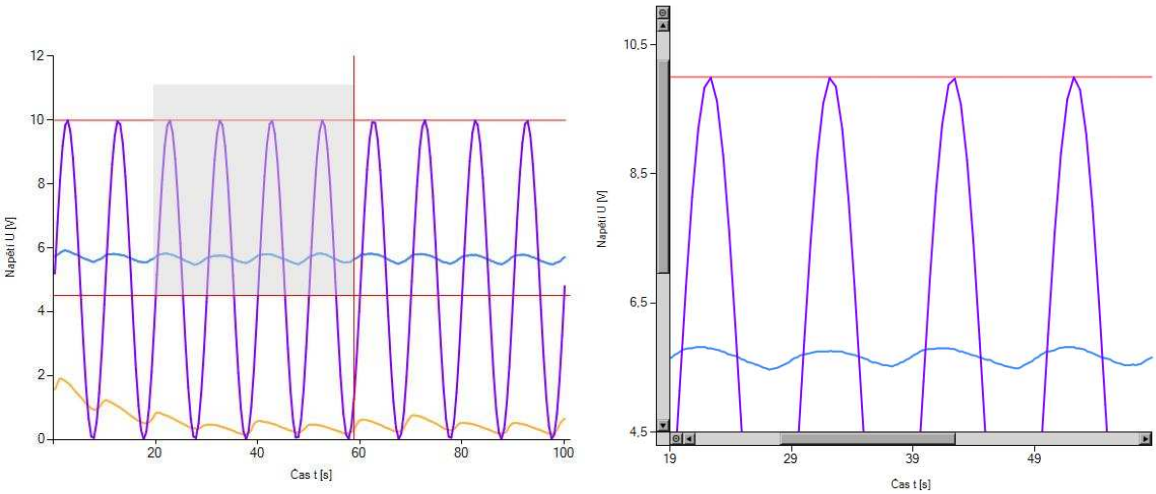


Obrázek 6-8 – Zobrazení měřených dat (a), parametry grafu (b)

U grafu lze nastavit tyto parametry (obrázek 6-8b):

- **Typ grafu**
  - **Line** – bodový s rovnými spojnicemi
  - **Spline** – bodový s vyhlazenými spojnicemi
  - **Stepline** – bodový se stupňovitými spojnicemi
  - **FastLine** – spojnicový pro rychlé vzorkování
  - **Point** - bodový
- 3D zobrazení grafu včetně možnosti rotace
- Zobrazení legendy
- Vložení titulku a jeho stylování
- Zobrazení bodů na křivce včetně zobrazení detailu (*tooltip*) při najetí myší na bod
- Tloušťka spojnic
- Zobrazení pomocné mřížky

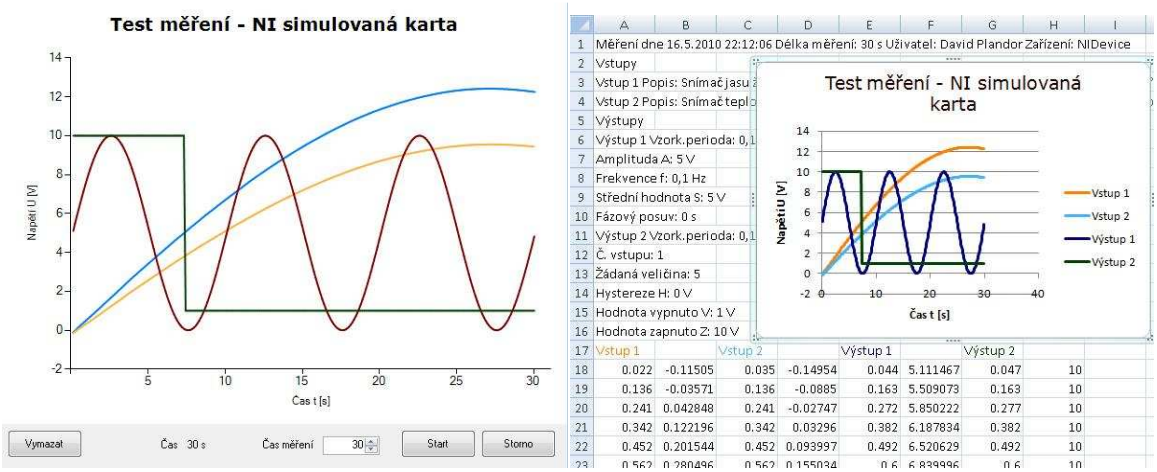
V grafu je možno použít také zvětšení detailu (zoom) přetažením dané oblasti myší se stisknutým levým tlačítkem (obrázek 6-9).



Obrázek 6-9 – Zvětšení detailu (zoom) v grafu

6.6 Export – import měřených dat a parametrů

Pro další zpracování dat slouží export do programu Microsoft Excel a to včetně textového popisu měření s uvedením údajů o měření – datum a čas měření, uživatel, který měření provedl a všech parametrů právě provedené regulace. V programu MS Excel se také vykreslí graf, který přebírá údaje jako je popis grafu nebo barvy datových řad z programu WinCoMeT.



Obrázek 6-10 – Měření a export měřených dat do programu MS Excel

Export do programu MS Excel využívá objekty OLE, je proto nutné mít tento program v počítači nainstalován. Uživatele, kteří tímto softwarem nedisponují, mohou využít export do formátu **.csv*, který obsahuje stejné údaje s výjimkou vykreslení grafu. Graf lze také exportovat přímo do grafických formátů **.bmp*, **.jpg*, **.png*, **.gif*, **.emf* a **.tif* nebo do schránky systému Windows, odkud je možno jej vložit do jiné aplikace (obrázek 6-8b).

Měřená data, akční veličiny regulátorů či generované signály lze v programu ukládat do specifického formátu **.cmt.* Z tohoto souboru je pak opět možno měření načíst a to včetně všech parametrů grafu a také veškeré použité parametry regulátorů a generovaných signálů. Soubor **.cmt* je v podstatě složený soubor formátu XML. Jedná se o exportovaná data z objektu grafu a navíc také všechny parametry aktuálního měření. Při importu zpět, tedy otevření souboru **.cmt*, jsou nejprve data vložena do objektu grafu a následně je vytvořen *dataset* s tabulkami a ty jsou přímo napojeny na *DataGridView*, která jsou v běžném režimu napojena na databázové tabulky. V aplikaci je velmi často využita konverze objektových dat do XML. Programový kód vypadá následovně:

```
MemoryStream ms = new MemoryStream();
XmlSerializer s1 = new XmlSerializer(typeof(DataSet));
s1.Serialize(ms, mDeviceParams);
string test = Encoding.UTF8.GetString(ms.ToArray());
```

Takto je z datasetu vytvořen text a ten je následně uložen do databáze. Je-li potřeba, provede se reverzní operace, z textového souboru se vytvoří přímo odpovídající objekt.

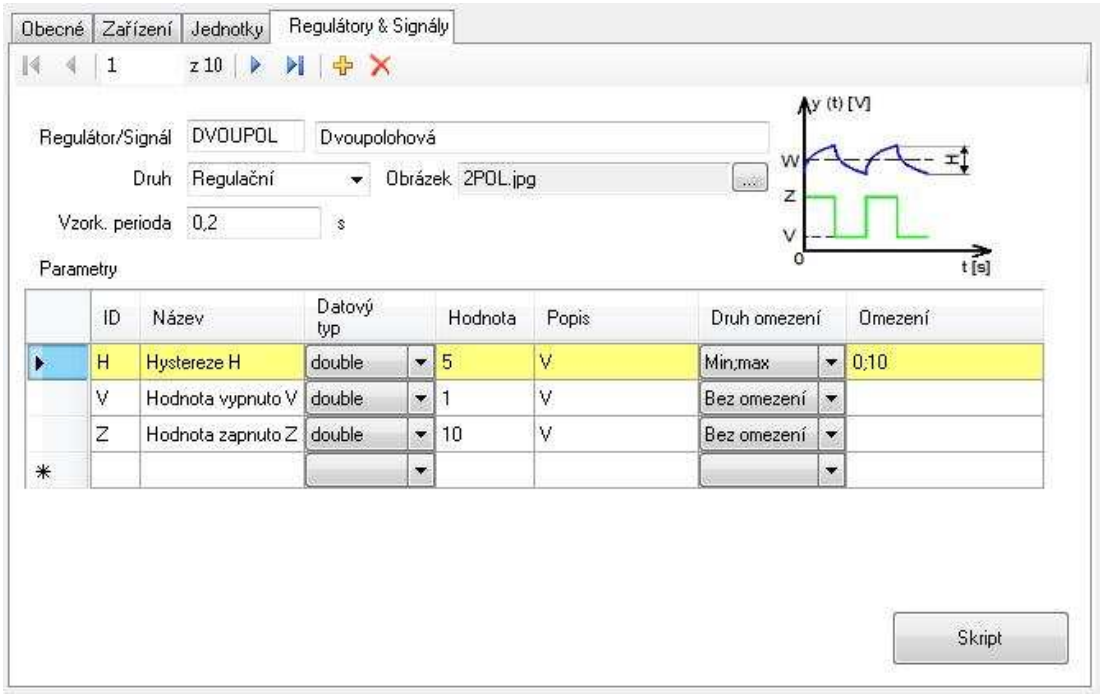
```
StringReader sr = new StringReader(dt.Rows[0]["DLL_PARAM"].ToString());
XmlTextReader r = new XmlTextReader(sr);
XmlSerializer s = new XmlSerializer(typeof(DataSet));
DataSet mDeviceParams = (DataSet)s.Deserialize(r);
r.Close();
```

Na podobném principu je založen také interní formát **.cmp*, do kterého lze ukládat uživatelská nastavení programu. Ze souboru lze zpět tato uživatelská data načíst.

6.7 Nová zařízení a regulátory

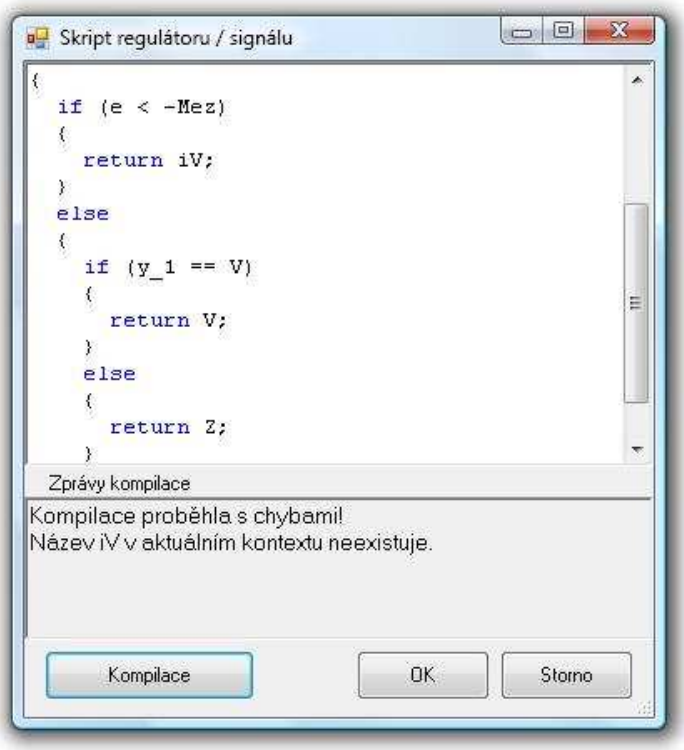
Nová zařízení lze do programu vložit v obecném nastavení. Stisknutím tlačítka *Nové zařízení* se otevře dialog pro výběr **.dll* souboru na disku počítače. Po výběru je daná komponenta verifikována a vyhovuje-li, je přidána do aplikace, jsou nadefinovány vstupy a výstupy zařízení, jsou zkompilovány její třídy a metody. Jsou do databáze přidány všechny potřebné záznamy o vstupech a výstupech. Pak lze již zařízení využívat.

V aplikaci je také možno definovat vlastní regulátory a signály a také editovat již existující záznamy. U všech parametrů lze nadefinovat datový typ parametru, implicitní hodnotu, druh omezení a vlastní omezení. Omezení spočívá v zadání intervalu, ve kterém musí zadaná hodnota existovat nebo také možnost zadání seznamu možných hodnot.



Obrázek 6-11 – Definice regulátorů a signálů

Zadaná omezení se kontrolují ve formuláři měření. Všechny tyto údaje jsou uloženy v databázi, a aby bylo možno je používat, je nutno je zkompilovat. Kompilace se provádí automaticky při ukládání záznamů. Při definici skriptu regulátoru / signálu může uživatel ověřit správnost programového kódu stisknutím tlačítka *Kompilace*, při výskytu chyb se zobrazí jejich seznam (obrázek 6-12).



Obrázek 6-12 – Kompilace skriptu regulátoru / signálu

Syntaxe skriptu vychází z jazyka C#. Pro matematické výpočty jako jsou např. goniometrické funkce nebo odmocnina je nutno využít třídu *System.Math*. Pro skript jsou k dispozici následující pevně dané proměnné. Jde o rezervované názvy proměnných, které se nedají uživatelsky definovat.

Tabulka 6-1 – Pevně definované proměnné při definici skriptu regulátoru (signálu)

Proměnná	Datový typ	Popis
w	double	Žádaná veličina
T	double	Vzorkovací perioda
t	double	Čas od začátku měření
e	double	Regulační odchylka
e_1	double	Minulá reg. odchylka
e_2	double	Předminulá reg. odchylka
u_1	double	Předchozí akční veličina

V aplikaci byly v rámci práce implementovány 3 regulátory a 6 generovaných signálů.

Regulátory

- Dvoupolohová regulace
- PID regulace
- Fuzzy PI

Generované signály

- Konstantní signál
- Obdélníkový signál
- Pravoúhlý impuls
- Trojúhelníkový signál
- Sinusový signál
- Šum

Z algoritmů zvolíme jeden pro ilustraci postupu při návrhu skriptu regulátorů a generovaných signálů. Vždy vycházíme ze vztahů pro daný regulátor a zapíšeme ve formě programového kódu jazyka C#. Zaměříme se na skript PSD (číslicového PID) regulátoru. Jedná se o přírůstkový algoritmus PSD regulátoru. Zápis algoritmu je následující. [Vítečková & Ščevík 2009]

$$u(kT) = u[(k-1)T] + q_0 e(kT) + q_1 e[(k-1)T] + q_2 e[(k-2)T] \qquad (5-1)$$

$$q_0 = k_p \left(1 + \frac{T}{T_I} + \frac{T_D}{T}\right)$$

$$q_1 = -k_p \left(1 + 2\frac{T_D}{T}\right) \qquad (5-2)$$

$$q_2 = k_p \frac{T_D}{T}$$

Dle těchto vztahů navrhneme výkonný programový kód. Ve skriptu je využita podobná symbolika, jako je v předchozích vztazích. Protože nelze při programování využít dolní indexy, jsou použita spojení dvou písmen a pro vyjádření předchozích regulačních odchylek podtržítka (k_p – zesílení regulátoru; T, T_I, T_D – vzorkovací perioda, integrační a derivační časová konstanta; u – akční veličina; k - krok).

```
double q2 = kp * Td / T;
double q1 = kp + 2 * q2;
double q0 = kp * (1 + T / Ti) + q2;
u = u_1 + q0 * e - q1 * e_1 + q2 * e_2;
return u;
```

7 Komponenty pro zařízení

Jak již bylo několikrát řečeno, komunikaci s hardwarovým zařízením a aplikací WinCoMeT zajišťují komponenty neboli *.*dll* knihovny. Tyto knihovny jsou vytvořeny opět v prostředí Visual Studio 2008. Aby bylo možné vytvářet knihovny pro nová zařízení, musí existovat konvence, kterou je nutné dodržet. Proto byla stanovena pravidla pro názvosloví tříd a metod, které jsou volány z aplikace WinCoMeT.

Po dynamické kompilaci komponenty jsou načteny hodnoty standardních proměnných.

Povinné proměnné jsou:

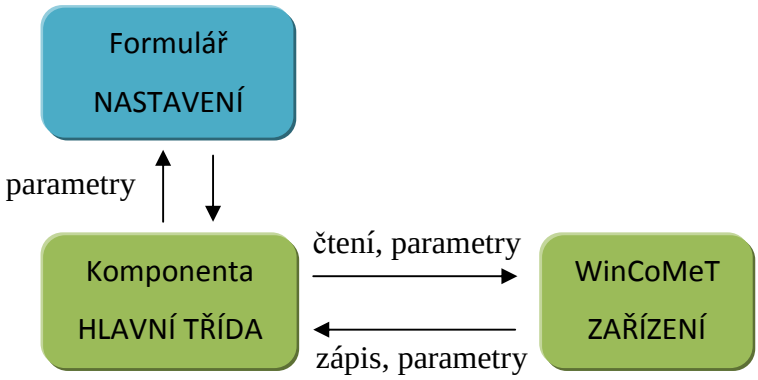
- celočíselná proměnná pro počet vstupů
  ```
  public int mInputsCount = 2; (pro 2 vstupy)
  ```
- celočíselná proměnná pro počet výstupů
  ```
  public int mOutputsCount = 2; (pro 2 výstupy)
  ```

Poté je vyhledána třída, která musí mít stejný název jako má daná komponenta, avšak nehraje roli velikost písma (*case sensitivity*). Je vytvořena instance této třídy. Třída má konstruktor s povinným parametrem typu *dataset*, který je určen pro parametry, které jsou uložené do databáze v aplikaci WinCoMeT. Výhoda tohoto řešení je okamžitě připravená komponenta pro použití včetně nastavení, které bylo provedeno během minulých měření. Pak je již možno volat metody, které musí být také standardní.

Jsou to tyto metody:

- metoda pro čtení hodnoty
  ```
  public double ReadValue(int channelNr)
  ```
- metoda pro zápis hodnoty
  ```
  public void WriteValue(int channelNr, double value)
  ```
- metoda pro nastavení
  ```
  public object Settings(DataSet param)
  ```

Vstupní proměnné těchto metod jsou číslo vstupu (*channelNr*) typu *integer*, zapisovaná hodnota (*value*) typu *double* a parametry (*param*) typu *dataset*. Parametry zařízení jsou nastavovány na straně komponenty. Komponenta má tedy vlastní formulář, kterým se její parametry nastavují. Tyto parametry jsou pak ve formě datasetu posílány zpět do aplikace WinCoMeT, kde jsou uloženy do databáze do tabulky zařízení.



Obrázek 7-1 – Komunikace aplikace WinCoMeT s komponentou

Při příštím vytváření instance jsou uložené parametry posílány do komponenty přímo, není proto nutné při každém spuštění aplikace znovu komponentu nastavovat. Na obrázku 7-1 vidíme systém komunikace třídy zařízení na straně aplikace WinCoMeT a hlavní třídy komponenty, která komunikuje s vlastním formulářem pro nastavení a předává si s ním parametry. Pro interní komunikaci uvnitř komponenty nejsou dána jakákoli pravidla, záleží tedy na individuálním přístupu autora komponenty, nutné je pouze předat ve výstupním parametru metody *Settings* proměnnou typu *dataset*, která může být v krajním případě i prázdná (*null*).

Vstupy a výstupy jsou interně číslovány od 0 do n-1, kde n je počet vstupů resp. výstupů. Tato čísla jsou posílána jako čísla vstupů a výstupů při čtení a zápisu hodnot. Jde o konvenci, kterou je třeba dodržet při návrhu komponenty. V komponentě je vhodné vytvořit proměnnou typu pole nebo seznam, pak číselné označení používat jako index pro zjištění označení vstupu charakteristického pro konkrétní komponentu. Programové řešení pro komponentu jednotky CTRL vypadá takto:

```
private string[] mInputs = new string[] { "A0", "A1", "A2", "A3" };
string channel = mInputs[channelNr];
```

Univerzální přístup nejlépe vysvětlí srovnání s řešením u komponenty pro zařízení od společnosti National Instruments.

```
public string[] mInputs  = new string[]{"Dev1/ai0", "Dev1/ai1"};
```

Obecně si tedy aplikace WinCoMeT vyžádá od komponenty hodnotu pro vstup 0 a tuto hodnotu obdrží. Jak se interně tento dotaz v komponentě řeší, ji nezajímá.

8 Srovnání aplikací

Nyní provedeme srovnání aplikace WinCTRL v2 a WinCoMeT. Nejvhodnější a nejnázornější bude srovnání formou tabulky.

Tabulka 8-1 – Srovnání aplikace WinCTRL a WinCoMeT

Vlastnost	WinCTRL v2	WinCoMeT
Nastavení sériové komunikace	COM1, COM2	COM porty přítomné v PC
Kompilace při změnách	Celý program	Jen program, jen komponenta
Podporovaná zařízení	Pouze jednotka CTRL	Zařízení pod MS Windows
Provoz bez hardwar. zařízení	Ne	Ano (simulace zařízení)
Podpora Unicode	Ne	Ano
Změna parametrů během měření	Žádaná veličina	Všechny
Velikost grafu při měření	Fixní	Variabilní
Změna parametrů grafu	Pouze některé parametry	Plně editovatelné
Počet exportovaných sérií grafu	Nejvýše 7	Neomezeně
Seznam regulátorů /signálů	Pevně daný	Uživatelsky rozšiřitelný
Ukládání parametrů uživatele	Některé	Všechny
Export dat	Interní formát, **.txt*	Interní formát, MS Excel,**.csv*
Export grafu	**.bmp*	6 formátů, schránka, MS Excel
Detail grafu (zoom)	Ne	Ano
Aktualizace programu	Neřeší	Zjišťuje z internetu

Z uvedených skutečností je jasné, že nová aplikace WinCoMeT je obecnější, dynamičtější, poskytuje širší možnosti exportu dat, paralelní přístup k datům a jejich grafickému zobrazení, zatímco poskytuje jednoduchost, na níž je orientována původní aplikace WinCTRL. V programovém kódu aplikace WinCoMeT se veškeré funkce vyskytují pouze na jednom místě a jsou logicky umístěny ve třídách a formulářích. Programový kód tímto získává přehlednost a ladění takovéto aplikace je snadnější. Srovnání obdobných měření v programu WinCTRL a WinCoMeT naleznete v příloze B. Záměrně je v této příloze ponechán surový formát grafů, tak jak je z aplikace získáme.

9 Závěr

Na základě zadání práce jsem nastudoval konkrétní problematiku teplovzdušného modelu a jednotky CTRL, které se využívají v laboratoři pro výuku regulace. V úvodu práce jsem tato hardwarová zařízení popsal. Zaměřil jsem se především na softwarovou část úlohy, neboť cílem bylo vytvořit aplikaci, která by aktuálně používaný program nahradila. Bylo potřeba provést důkladnou detailní analýzu programu WinCTRL, zaměřit se na její nedostatky a naopak zachovat funkcionality, které uživatelům vyhovují. Vzal jsem také v úvahu připomínky vedoucího práce a vyučujících z laboratoře. Dle těchto poznatků jsem vypracoval funkční a databázový model aplikace. Pak již bylo nutné zvolit vývojové prostředí a další potřebné nástroje. Z důvodu širokých možností bylo zvoleno vývojové prostředí Microsoft Visual Studio 2008 a platforma .NET Framework. Z programovacích jazyků nejlépe vyhovoval C#, neboť s jeho užitím již mám zkušenost. Pro návrh databázového modelu byl využit *Dataset Designer* prostředí Visual Studio, jako databázový systém velmi dobře posloužil systém Microsoft SQL Compact, který lze přímo implementovat do aplikace a není nutno jej instalovat zvlášť. Pro zobrazování měřených dat do grafu byla zvolena komponenta Microsoft Chart Controls, jež je pro platformu .NET Framework 3.5 navržena. Tato komponenta umožňuje zobrazovat data do velmi efektních grafů, navíc disponuje širokou škálou nastavení. Měřená data jsou také uchována v přehledných tabulkách a je možno k nim přistupovat paralelně se zobrazením v grafu. Výsledný graf je možno v programu dodatečně upravovat, přidávat popisy, měnit barvy, styly zobrazení datových řad atd. Změřená data a také uživatelské parametry programu je možno uložit do souboru interního formátu aplikace. Z tohoto souboru je pak možno opět tato data do programu načíst. Kromě toho lze měřená data exportovat do programu Microsoft Excel včetně uvedení popisných informací a vykreslení plně editovatelného grafu.

Silnou stránkou aplikace WinCoMeT je užití komponenty pro komunikaci s hardwarovým zařízením. Výhoda spočívá v oddělení prezentační části od komunikační řešené pomocí komponent. Vytvořením komponenty a jejím načtením do programu WinCoMeT lze bez zásahu do samotné aplikace využívat konkrétní hardwarové zařízení. Okruh zařízení, která je možno s programem WinCoMeT využít, tak není omezen. Podmínkou je pouze podpora systémů Microsoft Windows a možnost k nim programově přistupovat.

Nad rámec zadání této práce se podařilo realizovat systém definice regulátorů a generovaných signálů. Uživatel má možnost definovat vlastní regulátory či signály nebo může jejich existující výkonný programový kód editovat. Využívá se zde opět dynamické kompilace. Zápis výkonného skriptu je vytvořen jako obdoba vývojového prostředí, jednotlivá klíčová slova jsou charakteristicky obarvena. Parametry regulátorů a signálu lze typově definovat včetně omezení jejich uživatelem zadávaných hodnot.

Všechny tyto vlastnosti vytvářejí z programu WinCoMeT univerzální nástroj pro monitorování a regulaci procesů. Laboratorní teplovzdušný model s jednotkou CTRL je jednou z možných využití.

Součástí práce je vytvořená šablona pro vytvoření komponenty pro nová zařízení. Aplikace ověřuje při své inicializaci novější verze aplikace na internetu, což znamená pro uživatele větší komfort a snazší distribuci nových verzí. Pro další rozvoj navrhuji vytvořit jazykové mutace programu, především anglickou. Také by bylo vhodné navrhnout a realizovat algoritmus pro simulaci zařízení a komponentu pro něj. Takto by bylo možné používat aplikaci pro výuku regulace bez potřeby připojení k teplovzdušnému modelu. Velmi zajímavé by bylo také síťové řešení klient – server, kdy by se uživatel přes internet připojil k serveru s teplovzdušným modelem a prováděl by všechna měření vzdáleně ze svého počítače.

Aplikace WinCoMeT nabízí jednoduchý přehledný univerzální nástroj pro měření a regulaci, uživateli se základy programování nabízí velmi variabilní nástroj pro měření a experimenty s regulací. Doufám, že si tato aplikace najde své místo mezi dostupnými nástroji a nabídne odpovídající komfort všem zájemcům o pochopení principů regulace.

10 Použitá literatura

ALBAHARI, B., DRAYTON, P., MERRILL, B.: *C# Essentials.*, O'Reilly Media, 2002, 216 s., ISBN 0-596-00315-3.

BABIUCH M.: *Počítačové systémy.* VŠB - VŠB-TU Ostrava, Ostrava, 2007, 1. vydání, 242 stran. ISBN 978-80-248-1503-9.

BABIUCH M.: *Programování aplikací pro Internet II.* VŠB - VŠB-TU Ostrava, Ostrava, 2007, 1. vydání, 182 stran. ISBN 978-80-248-1504-6.

BABIUCH, M., HNIK, M.: *Using Technology of .NET Web Services in the area of Automation.* In Transactions of the VŠB - Technical University of Ostrava, Mechanical Series, No. 2/2009, volume LV, article No. 1680, p. 1-6, ISBN 978-80-248-2144-3, ISSN 1210-0471 (Print), ISSN 1804-0993 (Online), ISSN-L1210-0471.

BABIUCH M. & HRBÁČ D.: *Visual Studio .NET and VISIO (UML) Integration - Correspondence between source files and class models.* In Proceedings of 7th International Carpathian Control Conference. Rožnov pod Radhoštěm, Czech Republic, May 29-31, 2006, pp.33-36. ISBN 80-248-1066-2.

BABIUCH, M., LANDRYOVÁ, L. *Data Model in Industrial Automation Using New Technologies.* Sborník vědeckých prací VŠB-TU Ostrava, řada strojní r. LIV, 2008, č. 1612, s. 1-6. ISSN 1210-0471. ISBN 978-80-248-1870-2.

COOPER,D.: *Practical Process Control*, 2008, online, dostupné z www <URL: http://www.controlguru.com/pages/table.html>.

FARANA, R.: *Databáze – speciální postupy.* 1. vyd. Praha : Český svaz vědeckotechnických společností a další, 2006. 170 s. ISBN 80-02-01876-1.

FARANA, R. a kol.: *Doporučení pro psaní odborných textů z oblasti automatizace a informatiky.* 1. vyd. Ostrava : VŠB-TU Ostrava, 2008. 80 s. ISBN 978-80-248-1925-9.

FARANA, R.: *Tvorba relačních databázových systémů.* 1. vyd. Ostrava : VŠB-TU Ostrava, 1999. 100 s. ISBN 80-7078-706-6.

GANESH, A., *Developing MDI Applications in C#*, c-sharpcorner.com, 2002, online, dostupné z www <URL: http://www.c-sharpcorner.com/UploadFile/ggaganesh/DevelopingMDIAppplicationsinCSharp11272005225843PM/DevelopingMDIAppplicationsinCSharp.aspx>.

GOREV,A.: *MS Chart Control*, msdn.com, 2009, online, http://blogs.msdn.com/alexgor/

GROH, M.: *Creating Components in .NET*, msdn.com, 2002, online, dostupné z www <URL: http://msdn.microsoft.com/en-us/library/ms973807.aspx>.

HILYARD, J., TEILHET, S.: *C# 3.0 Cookbook.*, O'Reilly Media, 2007. 886 s., ISBN 0-596-51610-X.

KAČMÁŘ, D.: *Programujeme .NET aplikace ve Visual studiu .NET.*, Praha: 2001, 335 s. ISBN 80-7226-569-5.

KLÁN,P., HONC,D., JINDŘICH,J.: *Nová měřicí jednotka CTRL V3*, Praha, 2003, 8s.

LANDRYOVÁ, L., BABIUCH, M. *Modeling Objects of Industrial Applications*. In Handbook of Research on Social Dimensions of Semantic Technologies and Web Services. - Chapter XXXVI pp. 743-759, 1099 pages. Informatic Science Reference, Hershey, New York, IGI Global 2009. ISBN 978-1-60566-650-1 hardcover, ISBN 978-1-60566-651-8 eBook.

MICROSOFT: *Chart Controls for .NET Framework*, 2008, msdn.com, online, dostupné z www <URL: http://social.msdn.microsoft.com/Forums/en-US/MSWinWebChart/threads/>.

MICROSOFT: *Design Guidelines for Class Library Developers*, 2008, msdn.com, online, dostupné z www <URL: http://msdn.microsoft.com/en-us/library/czefa0ke%28vs.71%29.aspx>.

MICROSOFT: *Jak automatizovat aplikaci Microsoft Excel pomocí jazyka Microsoft Visual C# .NET*, 2007, online, dostupné z www <URL: http://support.microsoft.com/kb/302084>.

MICROSOFT: *Samples Environment for Microsoft Chart Controls*, msdn.com, online, dostupné z www <URL: http://code.msdn.microsoft.com/mschart>.

MICROSOFT, *SQL Server Compact*, 2008, online, dostupné z www <URL: http://www.microsoft.com/Sqlserver/2005/en/us/compact.aspx.>.

NATIONAL INSTRUMENTS: NI-DAQmx 9.0.2 Readme, 2009, online, dostupné z www <URL:http://ftp.ni.com/support/softlib/multifunction_daq/nidaqmx/9.0.2/readme.html>.

PROSISE, J.: *Programování v Microsoft .NET*, Computer Press, 2003, 736 s., ISBN 80-7226-879-1.

PUŠ, P., *Poznáváme C# a Microsoft .NET*, 2006, online, dostupné z www <URL: http://www.zive.cz/Programovani/C_CSHARP/sc-74/default.aspx?section=74>.

SADI,S.: *Dynamic Method Invocation in C# .Net*, 2008, online, dostupné z www <URL: http://sadi02.wordpress.com/2008/05/14/dynamic-static-and-non-static-method-invocation-in-c/>.

SMUTNÝ, L.: *Řízení teplovzdušného modelu TVM pomocí PC a mikropočítačové jednotky CTRL - Návod k laboratorní úloze.*, Ostrava, 2007, 17 s.

SSW CONSULTING SERVICES: *SSW .NET Object Naming Standard*, 2008, online, dostupné z www <URL: http://www.ssw.com.au/ssw/Standards/DeveloperdotNET/DotNetStandard_ObjectNaming.aspx>.

TROELSEN, A. W., *Pro C# 2008 and the .NET 3.5 platform*, New York, Apress, 2007, 1340 s., ISBN 1-59059-884-9.

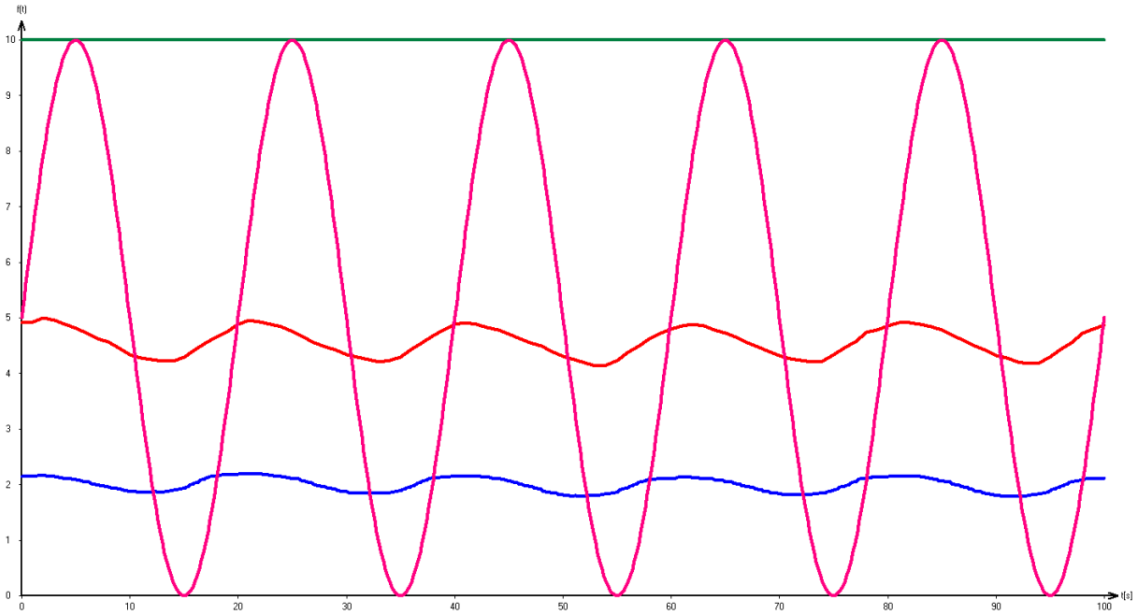
VÍTEČKOVÁ, M., ŠČEVÍK,P.: *Číslicová regulace*, online, 2009, dostupné z www <http://www.fs.vsb.cz/books/cislicovaregulace/>.

11 Přílohy

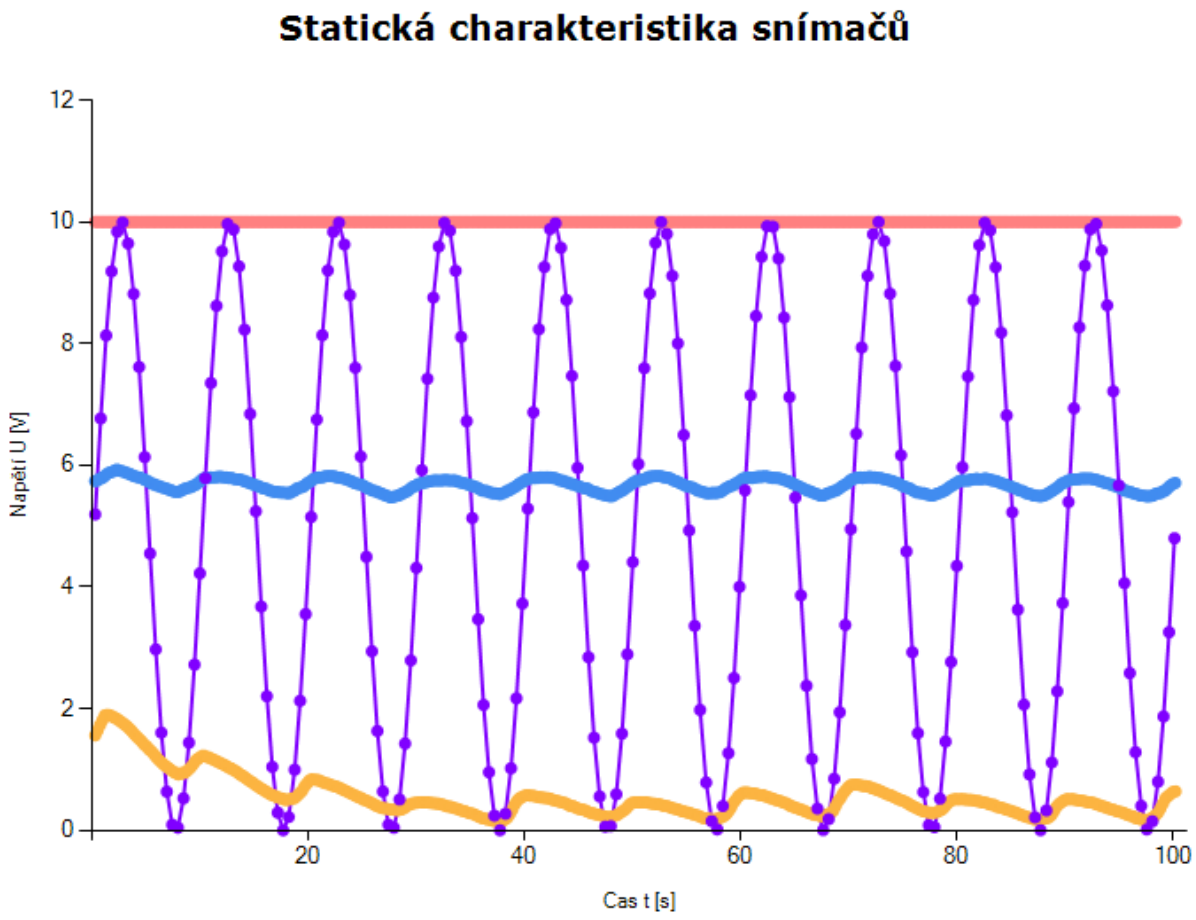
Příloha A – Zapojení pinů konektoru jednotky CTRL

PIN	Signál konektoru	Signál svorkovnice
1	Analogový vstup 0 (0-10V)	A0
2	Analogový vstup 1 (0-10V)	A1
3	Analogový vstup 2 (0-10V)	A2
4	Analogový vstup 3 (0-10V)	A3
5	nezapojen	-
6	V - (napájení 12V nestab.)	VE
7	GND (společná signálová zem)	0V
8	GND (společná signálová zem)	0V
9	GND (společná signálová zem)	0V
10	Digitální vstup 3	I3
11	Digitální vstup 2	I2
12	Digitální vstup 1	I1
13	Digitální vstup 0	I0
14	V+ (napájení 12V nestab.)	V+
15	Digitální výstup 0	S0
16	Digitální výstup 1	S1
17	Digitální výstup 2	S2
18	Digitální výstup 3	S3
19	Analogový výstup 0 (0-10V, 50mA)	N0
20	Analogový výstup 1 (0-10V, 50mA)	N1
21	+5V (stabil.)	5V
22	GND	0V
23	GND	0V
24	GND	0V
25	GND	0V

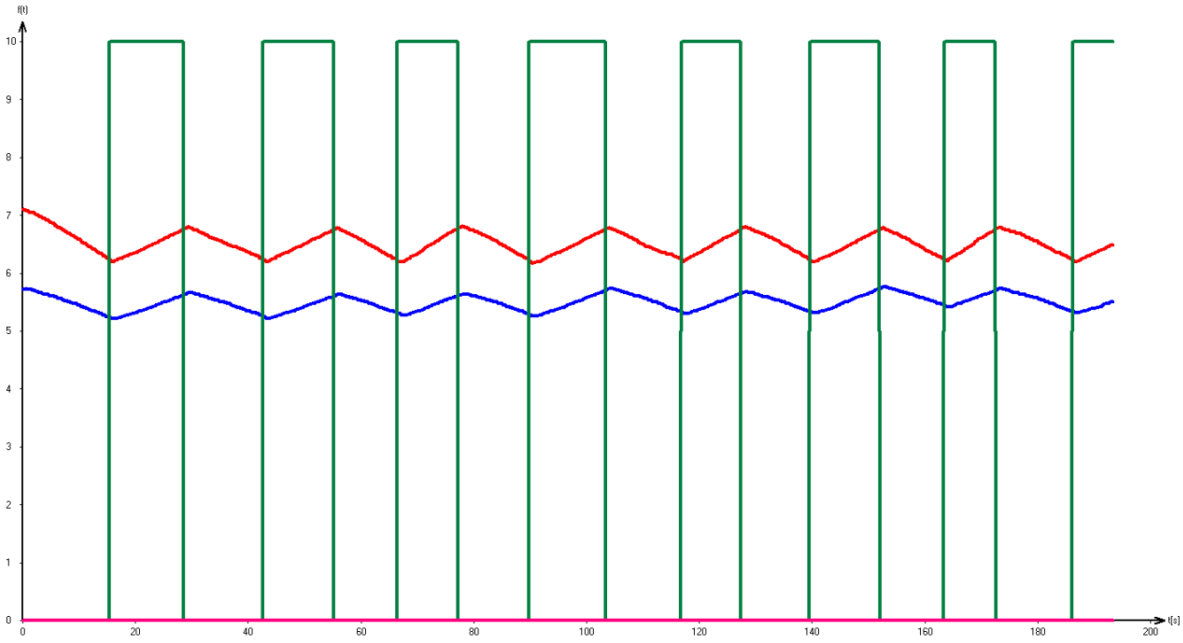
Příloha B – Srovnání obdobných měření ve WinCTRL a WinCoMeT



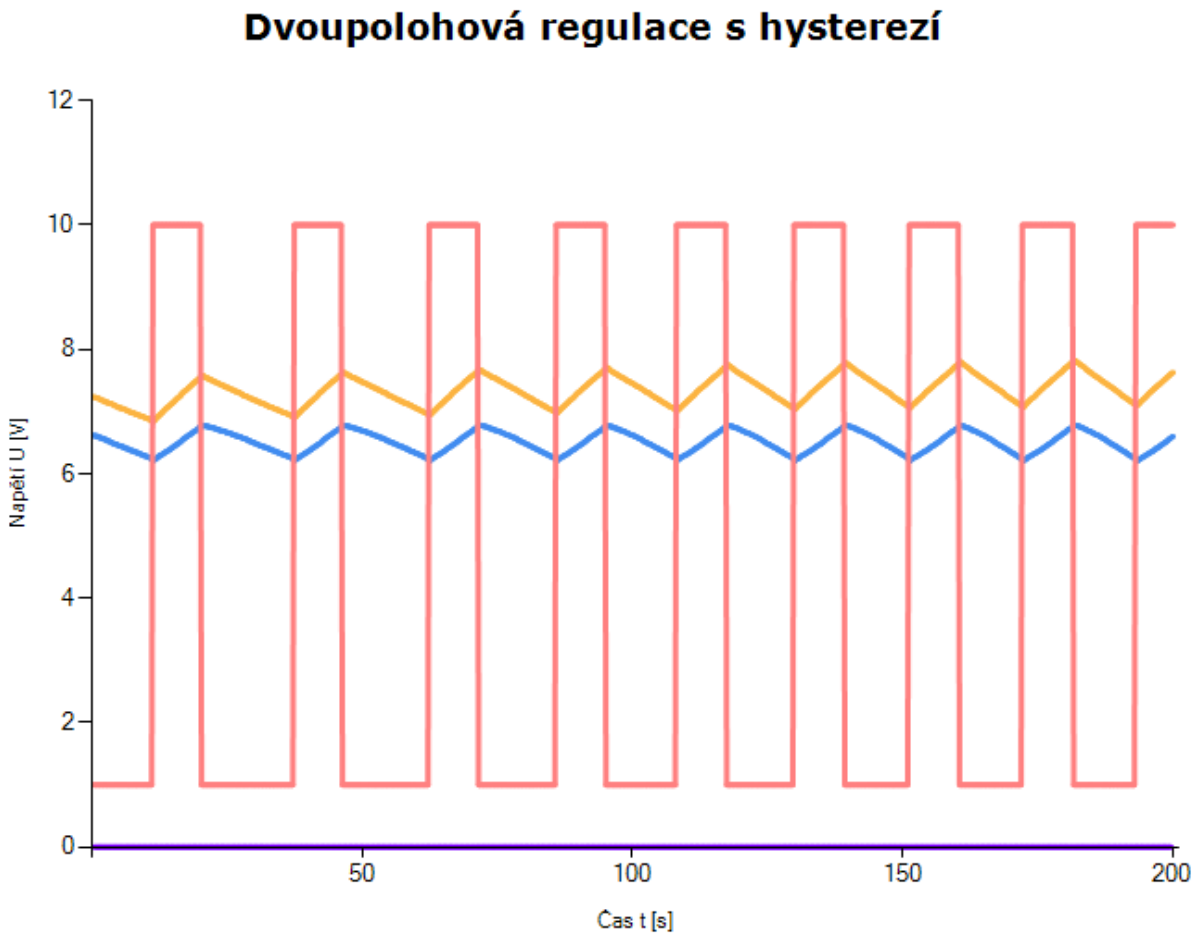
Graf statické charakteristiky snímačů exportovaný z aplikace WinCTRL



Graf statické charakteristiky snímačů exportovaný z aplikace WinCoMeT (se zobrazením bodů)



Graf dvoupolohové regulace exportovaný z aplikace WinCTRL



Graf dvoupolohové regulace exportovaný z aplikace WinCoMeT